1998

The study, design, and implementation of Data mart functions in Windows environments

Shenning Wen

THE STUDY, DESIGN, AND IMPLEMENTATION OF DATA MART FUNCTIONS IN WINDOWS ENVIRONMENTS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Shenning Wen

March 1998

THE STUDY, DESIGN, AND IMPLEMENTATION OF DATA MART FUNCTIONS IN WINDOWS ENVIRONMENTS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Shenning Wen

March 1998

Approved by:

[redacted] 3/17/98

Dr. Tong L. Yu, Chair, Computer Science Date

[redacted]

Dr. Richard J. Botting

[redacted]

Dr. Josephine Mendoza

ABSTRACT

Like other businesses in industry, there is a need to have a software system to give managers and other decision-making executives easy access to up-to-date operating information. This information includes inventory, bookkeeping, and product maintenance activities between central/headquarters and branch stores. The traditional way to provide information to those who make critical decisions is to build a centralized data warehouse utilizing various legacy systems.

This project intends to explore the feasibility of using a data mart[1] structure to expedite the decision-making process and to implement key functions of such a design. This would provide users with critical, timely, and accessible decision-making information. The task of this project is to build the communication channel within the enterprise management territory and to provide crucial marketing business intelligence in a timely fashion.

The purpose of this project is to develop a Decision Making Database System(DMDS) that utilizes Object Linking and Embedding(OLE) technologies to simulate how to integrate existing data from many different formats and locations into

[1]A **data mart** can be regarded as a special kind of data warehouse in which a summarized, highly focused portion of

a form that supports the decision-makers' community. The DMDS was designed using Data Mart Information Packaging Methodology's refinement technology. Such methodology began with information package diagrams, evolving into star schemata, and then into the final physical implementation phase, using Visual Basic version 4.0.

The DMDS was developed including fully functional user-friendly screens in Windows 95 operating environments and is readily expandable to the Windows NT workstation platform if needed. Several Structure Query Language(SQL) queries were embedded in the DMDS for delivering fast, effective, and comprehensive business solutions. The DMDS also assists users, who gain a significant competitive advantage by being shown a final visualization of data through information-filled reports, eventually assimilating enough information to make an informed decision.

The functionality of DMDS was fully validated and is now ready to deploy on the intranet of any of today's business corporations.

---

the data is placed into a separate database for use by a specific population.

ACKNOWLEDGEMENTS

First of all, I would like to thank my project committee professors: Dr. Yu, Dr. Botting, and Dr. Mendoza for their strong support in developing this project. Their valuable guidance and suggestions contributed substantially to this project.

And thanks to Tom Hammergren the author of *Data Warehousing: Building the Corporate Knowledge Base*, for giving me a solid base to build upon. Mr. Hammergren's Information Packaging Methodology is adapted, and assimilated in chapters 4, 5, 6 of this project report.

I would like to acknowledge Jonathan Anderson and Adam Hartmann for their enthusiasm and feedback, which helped me improve the text of the project report. Thanks also are due to Associated Studies, Incorporated (ASI), for their grant to provide partial funding support for this project.

Finally, a special note of appreciation to my family members-in particular, my mother, whose encouragement and patience made my education possible.

Shenning Wen

To Chu-Shuan, My Mother

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

The motivation of this project comes from the idea of finding a way to construct a better, faster, and more complete information system, which provides for the needs of the user.[1, 2] A similar vision is echoed by MS Solutions, the Riverside, California-based retail software provider, which assists clients in becoming more profitable and more competent in the current competitive retail market[3]. Like other businesses in the software industry, MS Solutions needs to render a software system, which will give managers and other decision-making executives easy access to the most up-to-date operating information.

One of the existing difficulties worth mentioning in the retail business is the transference of data or diagrams within different platforms, sets of peripherals, or those of incompatible applications. One of the most frequently asked questions by retail software developers is whether it is feasible to allow such a transfer to occur naturally in different application environments. The Object Linking and Embedding (OLE) techniques, [10, 11, 20] which are mainly used in Windows 95 and Windows NT platforms, provide a solution to this question.

The techniques of OLE not only enable users to integrate applications and share data more easily on a system, but also reduce hardware and training costs, and comply with the current trend of downsizing in the management of human resources.

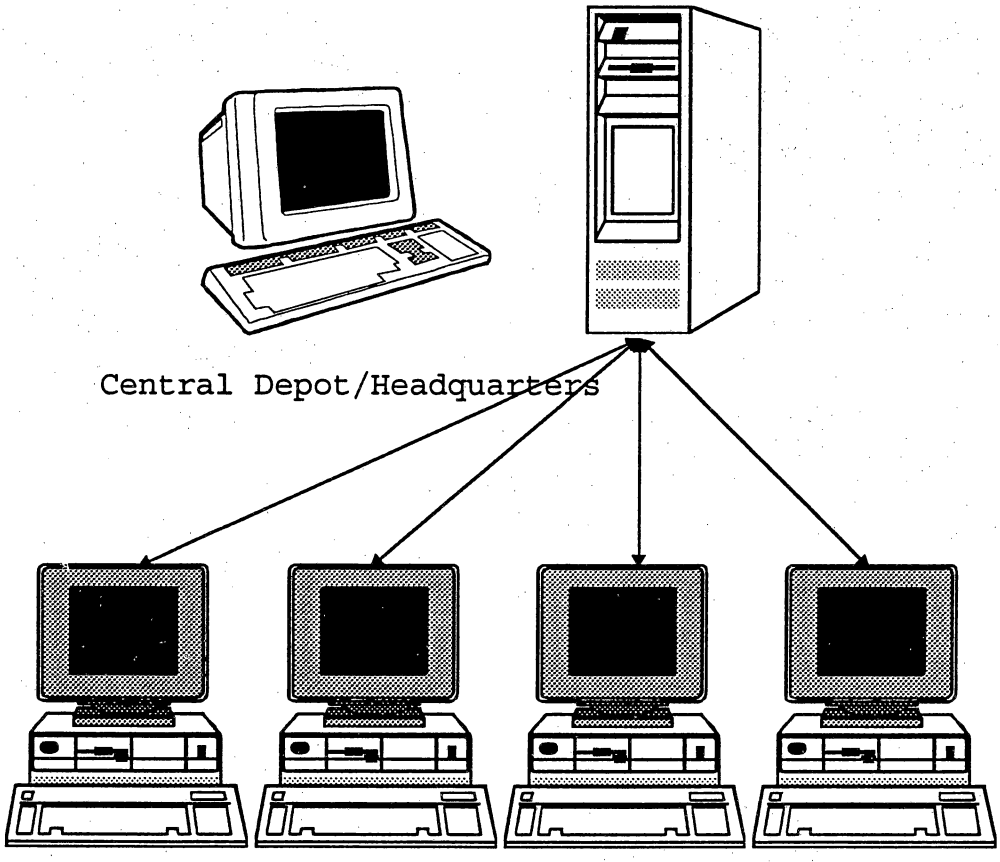


Figure 1.1. The design for the need of data communication between headquarters and its branch stores.

The traditional way to provide information to decision-makers is to build a centralized data warehouse, utilizing various legacy systems to make critical decisions.[3, 4, 5, 6, 7, 8]

Although this serves the need, according to a recent International Data Corporation study,[5] the average return on investment (ROI) in data warehouse technology is 321%, and the average payback time is 2.73 years. On the other hand, the study also shows that the average ROI for the data mart technology discussed below is 532%, and 1.57 years for the average time of payback.

A data mart can be regarded as a special kind of data warehouse, in which a summarized and highly focused portion of the data is placed into a separate database for use by a specific population.[1, 3, 4, 5] In other words, a data mart is a highly centralized data warehouse, catering to retail businesses' decision-makers.

Therefore, the main objective of this project is to explore the feasibility of using the data mart structure to expedite the decision-making process and implement some crucial functions of the data mart paradigm. This would provide critical, timely, and accessible decision making information to the decision-makers.

PURPOSE OF THIS PROJECT

The purpose of this project is to develop a Decision Making Database System (DMDS) and utilize OLE technology to provide an effective tool to update Point-Of-Sale (POS) transaction results among the headquarters and branch stores

on any business day. (Figure 1.1) This project would render some timely and crucial marketing information for the decision-makers, such as the central station manager having full control of general operations. These operations include inventory checking, purchase order control, price setting, order delivery information, and daily, weekly, and annual product movement analysis. The final goal is to provide what is needed for the decision, no more and no less.

PROJECT PRODUCTS

The completed project design will deliver the following products in the final phase:

- **Application of Decision Making Database System(DMDS)**

The name of the Database System speaks for itself. This system is built in such a manner that once its functions are successfully implemented within the enterprise management territory (i.e. Intranet), it will provide crucial marketing information in a timely fashion.

Visual Basic for Windows 95 (version 4.0), the coding platform language to implement this design, is well known as a great tool for creating robust and useful Windows-based applications. Its productivity-enhancing tools for graphical user interface (GUI) techniques are widely used throughout the system.

Visual Basic is also equipped with data-bound controls and has a powerful Jet Database Engine (the same database engine used by Microsoft Access in this project; for the detail of Jet Database Engine; please see Appendix B), which allows one to write Structured Query Language (SQL) statements to set up the database and front-end interfaces for the DMDS. The other advantage of using the Jet database engine in the project is that it accepts not only the native Jet database data but also the data from other external databases, such as dBase, Paradox, FoxPro, Btrieve, Lotus 1-2-3,or Excel.[9, 10] This allows the DMDS to house other systems' information easier. (illustrated in Figure 1.2 of the following page)

Microsoft OLE technology, which was used in one of the feature modules of this system, provides a standard way of supporting multiple retail POS platforms. This leads to the DMDS having higher integrity and productivity.

- **System Users Guide**:

Its function is to provide the best available reference document (this report) for the design detail, implementation specification, system requirements, and project limitations of the Decision Making Database System. The information in this guide will be used in an essential, up-to-date,

simplified jargon format to support the administration users who may or may not have a database background.

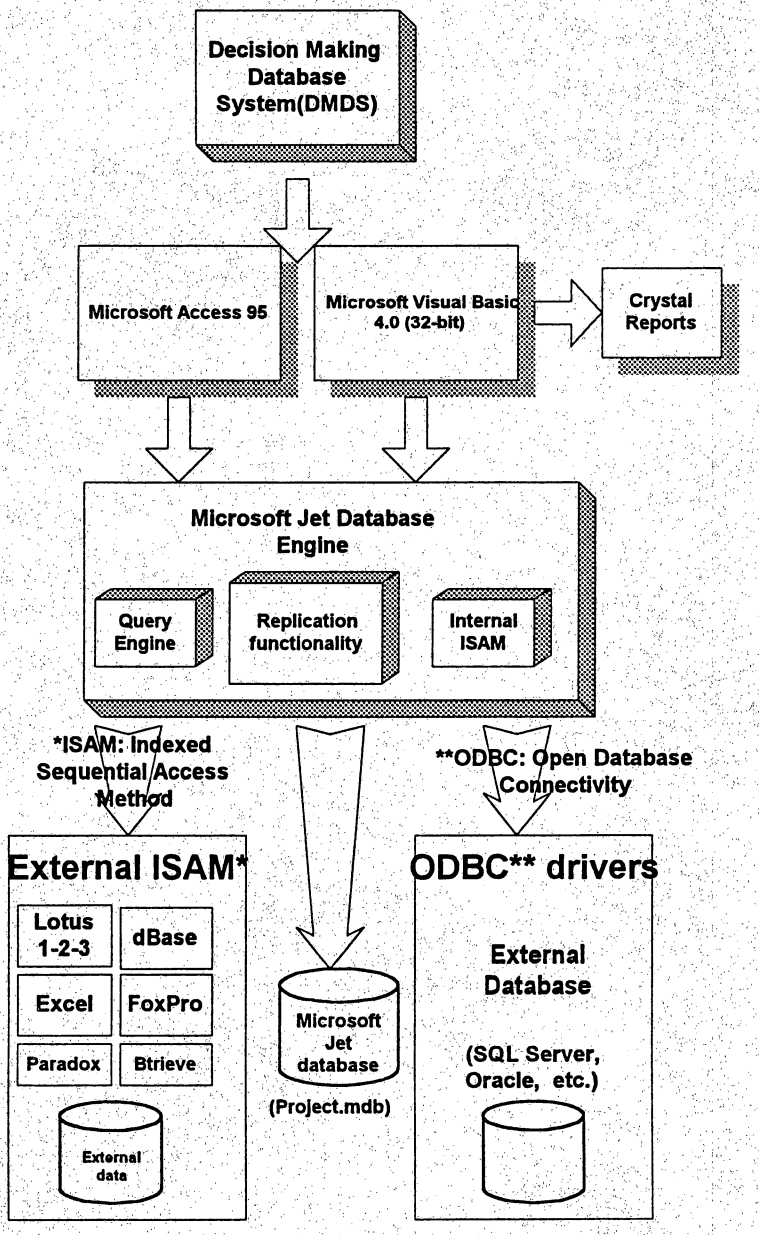


Figure 1.2 The structure of Decision Making Database System(DMDS)

CHAPTER TWO

SYSTEM REQUIREMENTS SPECIFICATION

INTRODUCTION

The mission of this project includes the design and implementation of key data mart functions. This project will show how to integrate existing data from many different formats and locations into a form that will support decision-makers in a timely fashion. The completely designed database system will offer corporate managers and other decision-making executives easy access to up-to-date operating information on inventory, bookkeeping, and product maintenance activities. The Visual Basic for Windows 95(version 4.0) enterprise edition is the coding platform language to implement this design.

PROJECT APPROACH

Current software systems such as *Retail*, developed by MS Solutions, pose the question of how to effectively integrate POS daily transaction data or seasonal promotion diagrams within various operating platforms or incompatible applications.

Historically, POS applications have been built in a low-level, hard-to-manage programming style (for example,

those that are used in NCR systems).[1, 8] Under these circumstances, the necessity to use high-level, powerful development tools (such as Microsoft Visual Basic) and cutting-edge technology (i.e., OLE for retail POS) to build a new system is inevitable; this is the point at which the project will be introduced.

The project is conducted using the concepts of a data mart paradigm[11], and the system design is constructed using the following steps:

1. Analyze the simulated operating environments, and clarify the scope of this project.
2. Review and understand the model requirements, and identify the key functions of the model in detail.
3. Collect or set up the testing data set for the project-testing phase.
4. Design the prototype by testing its level; this includes the following activities:
   - Identify entities within each testing scope.
   - Determine key attributes of each set.
   - Establish relationships.
5. Implement the design using Visual Basic as the coding platform language.
6. Test and evaluate the performance of the design module.
7. Analyze the performance and resource requirement of the implemented system.

PROJECT COMPONENTS

Like other nontrivial Windows applications, the project itself presents more than one screen. Each screen consists of objects that allow the user to interact with the systems in order to determine or manipulate the flow of the program. Generally speaking, the project can be categorized into two main components: modules and forms with controls. Each of these components has its unique functionality and appearance to contribute to the project:

- **Forms with controls:**

A form is a graphic user interface, which consists of a number of objects. Every object is made up of a variety of Visual Basic controls/tools box. In terms of functionality, every object can work as data input, results display, simply lead to another form of graphic / tabular representations, etc. These objects can have a diversity of representations, which include menus, command buttons, list boxes, drop-down combos, check boxes, data controls, and more.

Each object, once it is initiated by the user or system-generated events, will exhibit its own behavior through predefined programming code action.

- **Modules(Standard code modules):**

A code module contains common codes such as declarations, sub procedures, and functions that can share

vast levels of scope throughout the entire project application.

The big advantage of using standard code modules in the project is to eliminate the amount of repetitive codes and increase the overall modularity. In this project, one of the standard code modules, *graphlib*, contains a programmer-defined procedure that responds to a specific event to perform drawing functions like a 3-D pie chart, a 3-D bar chart, a line chart, and/or an area chart. Such modularity, therefore greatly enhances the power in the development of the project, and ensures the most efficient use of system resources.

FUNCTIONAL PARTITIONING

Just like a world full of objects and events, the concepts behind the project - object-oriented, event-driven programming (OOED), are easy to apply in order to design a user-friendly Graphic User Interface (GUI) application. The steps to create such an application using the OOED principle and language of Visual Basic are as follows:

- Create a new form and modify the properties of the form as desired.
- Construct the objects by picking up and dropping all the applicable customer controls from the

Visual Basic toolbox to be displayed on the newly created form.

- Write and attach customer codes to the target object's event, in response to user or system-generated occurrences.

Within a form, each object works as a functional unit. This can be illustrated by user-generated events in the forms of: a key is pressed, a key is released, a mouse click, a double mouse click, dragging and dropping, a selection from the pop up menu, or a specific button chosen from a dialog box, and so on. In addition, an event may be initiated by system occurrences such as form activation, a form load event, or the system timer checking at user-defined intervals.

However, if one looks at the Decision Making Database System (DMDS) from the application point of view, the functionality can be delineated by the choice within the user menu. This user menu serves as a gateway to link to a particular form to provide crucial, immediate information to satisfy the need of that moment. The *Select-an-Operation* form, which is one of the forms within the Decision Making Database System, is a good example that falls into this category.

In short, the goal of such a functional design echoes the need to provide a familiar and natural setting for the

Windows' environment as well as accurate information in an efficient manner for the data mart users.

FUNCTIONAL DESCRIPTION

- **Processing Scenario**

The nightly processing operation (NPO) - a default daily update process - provides the vital daily information for decision-makers allowing them to analyze and execute important tasks. These tasks include balancing product inventory, tracking product movement and pricing, updating certain pre-defined queries operations, or carrying out some dynamic queries from all stores.

Lastly, in order to enhance the company's capabilities for fast-paced growth and revenue expansion, the results of NPO can also be finalized in the form of a summary transaction report.

- **Restrictions/Limitations**

To run the Decision Making Database System, the system configuration requirements are as follows:

- PC with a 486DX/66 MHz or higher processor (Pentium or higher processor recommended).

- Windows 95 operating system, Windows NT Workstation operating system version 4.0, or Windows NT Workstation operating system version 3.51.
- 12 MB of RAM for Windows 95 or Windows NT Workstation 4.0(16 MB recommended); 24 MB for Windows NT Workstation 3.51.
- 15 MB hard-disk space required for minimum installation.
- VGA or high-resolution (800*600 minimum) monitor.
- Microsoft Mouse or compatible pointing device.

- **Performance Requirements**

The sample database file, *Project.mdb,* which is provided with the project as the testing data set, is a relatively small set of data if compared to the maximum size limitations. [10]Each form, standard, and class module of the Visual Basic database engine (Jet) can handle up to 64K of data. Since there are no explicit performance requirements set up for this application, the system response time for the calculations and displays of such a testing data set is pleasantly acceptable.

As more features are added to the application, the amount of memory the application uses and its executing speed will affect performance more significantly.

The following factors may, however, slow down the performance index in a significant way:[13]

- **Using vague data types for calculations**

  The default data type, such as Variant, consumes more system memory than the other data types. If frequently-applied mathematical computations were involved in the code, more precisely-defined variable data types (such as Integer or Long) will result in a considerable improvement in speed.

- **Using too many bitmaps and inappropriate formats to display on the application**

  Because an application such as this Decision Making Database System (DMDS) runs in a GUI environment, the application speed is greatly dependent upon the number of graphic bitmaps and proper display formats. The use of less graphic bitmaps and image customer control (compared to the use of picture box customer control) will lead to faster form and paint appearance, and therefore will have a positive impact on the application's performance.

There are some implicit requirements for this system design:

User-friendly

- It required the shortest possible time for the user to get used to the system.
- All designs are graphically illustrated; these include, for instance, pull-down menus and pop-up self-instructed help dialog boxes.

Efficiency

- Since this is the system to provide decision-making information, speed is very important for the user. Because the size of tested data is small to medium scale, the timing of the process response is above the acceptable range.

Reliability

- Once the event occurs, the user expects the system to behave in a stable, reliable and predictable manner.
- Results must not mislead the decision-maker.

Compatibility

- While this design is developed in Visual Basic (32-bit version) for a Windows 95 platform, it also can safely extend its ability to other operating systems, such as the Windows NT operating system environment.

Testability

- Each requirement in this design will be verified and thoroughly tested in the final product.

Security

- Only authorized personnel will have the privilege of accessing this database system.

VALIDATION CRITERIA

Prior to being put into practice, the database system will meet all of the requirements stated in the previous section. Any performance problems resulting from other existing programs outside this design scope will be disregarded in the evaluation stage.

CLASSES OF TESTS[14]

- **Unit Test**

  Test the individual form module in this design to ensure that the implementation functions perform as desired.

- **Integration Test**

  Integration testing is a systematic technique for assembling software while at the same time conducting tests to uncover errors associated with interfacing[9].

- **System Test**

  Test system capabilities under extreme cases to ensure that all the system elements have been properly integrated and will perform their allocated functions.

CHAPTER THREE

PROJECT APPROACH

One current strategy in decision-making systems software, such as *Retail* (developed by MS Solutions), is to build a centralized data warehouse structure utilizing various operational systems.[3, 4, 5, 6, 7, 8] This data structure provides a means for the users to make some critical decisions. According to recent research[5], using a new, emerging technology of database structure -- data mart -- has a more impressive and promising future for meeting most companies' information system department requirements.

A data mart, that is, a special kind of data warehouse,[7] has the following characteristics:[11]

- It provides a centralization of corporate data or information assets.
- It is contained in a well-managed environment.
- It has consistent and repeatable processes defined for loading operational data.
- It is built on an open and scalable architecture that will handle future data expansion.
- It provides tools that allow its users to effectively process the data into information without a high degree of technical support.

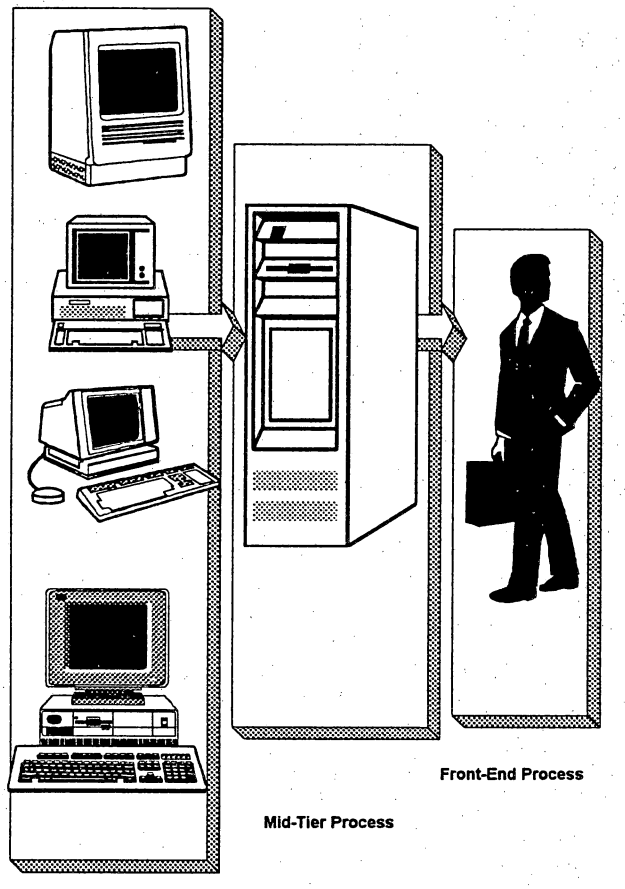


Figure 3.1 Components of a Data Mart Process

[11]There are three structural processes to make up a data mart system: Back End Process, Mid-Tier Process, and Front End Process (illustrated in Figure 3-1 of the previous page) The details of these processes are explained as follows:

- **Back End Process**: The main task of this stage is data collection and gathering. One of two major difficulties during the process is collecting a solid and consistent design specification from the uncertain requesters. The other impediment to muddle the entire data mart system is dealing with multiple source data formats. Some of them may be easy to manipulate; however, most may not, so the OLE technology is coming to play an important role in this project.
- **Mid-Tier Process**: After the Back End Process ends, the data need to be scrubbed to have uniform format and definition in order to be placed within the data mart system. This placement can be either in a central location (e.g. Headquarters) and/or in some remote locations (e.g. branch stores). Upon completing the data placement step, the standard reports that are contained within the data mart system should be compiled and indexed for the on-line users' information retrieval.

- **Front End Process**: This process involves granting the proper access levels to suitable users for retrieving the data mart system information. In this project, using the different security levels assigned to the distinct users to access the diverse DMDS data levels is a good practice of this concept. However, several critical user interfaces need to be built for ease of use. This is also a system developers' primary concern for this stage of design.

Although today's businesses are beginning to focus on how information can improve the way they do business, industry currently lacks a uniform methodology for capturing, transforming, building, and delivering such data into a data mart.[6]

This project, using Visual Basic and Access as implementing dialects, provides a sample of design and modeling techniques and delivers tools for the data mart paradigm. Since Visual Basic shares its database engine (Jet) with Access, databases created with Access can be managed through Visual Basic.[9, 10] Therefore, it is natural to select Access as a back-end support database for this project.

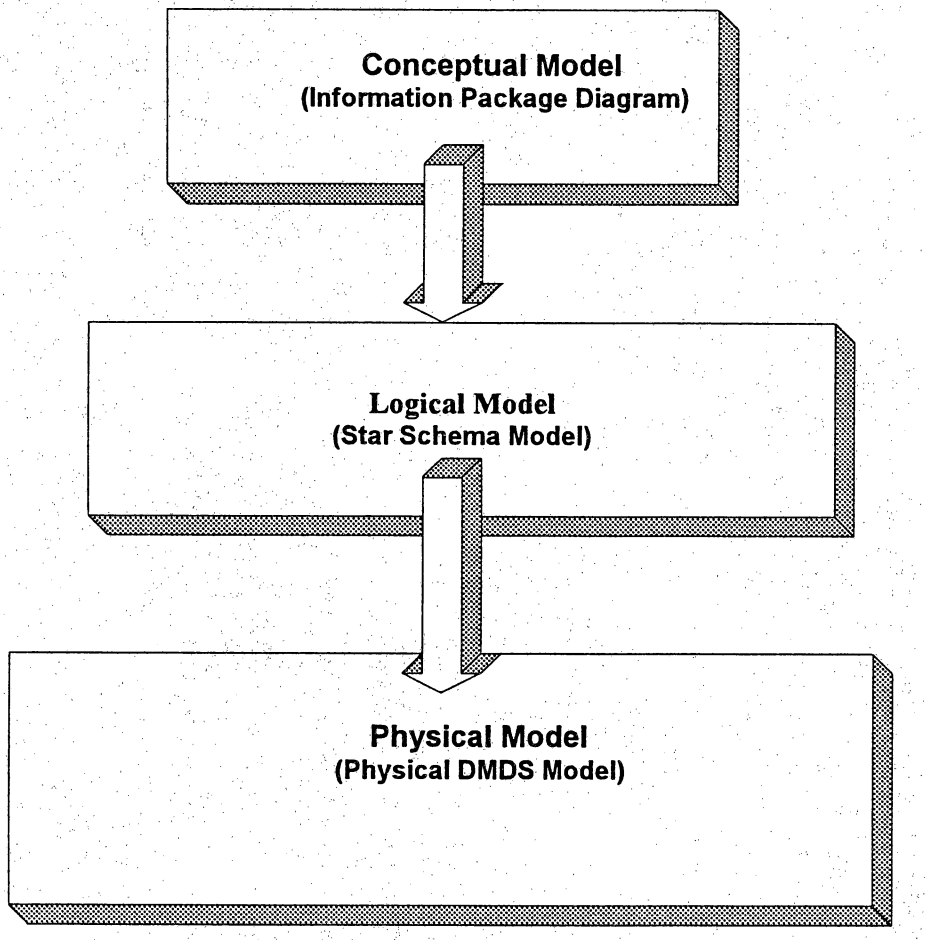


Figure 3.2 Data Model Of Decision Making DataBase System(DMDS)

In general, the design process of this project consists of three main phases.[6] The beginning phase involves the conceptual design of the system with information package diagrams. In the second phase, the star schema model will be built; the three logical entities comprising: measure, dimension, and category detail, will be defined and characterized. Last but not least, the final phase will

consist of physical design activities and construct the application front-end interfaces. A summary of these phases is provided below.

CONCEPTUAL MODEL DESIGN PHASE

A good data mart design depends strongly on providing its target users with a higher degree of satisfaction. These requirements will be properly defined in this phase design through a consistent, coherent communication tool - the information package diagram. The activity involved to build up this diagram for conceptual model design includes the following components:[11]

- Gathering requirements(usually referred to as information or dimension mapping) from the users as the basis for the overall design.
- Modeling and analyzing information requirements.
- Defining key performance measures.
- Defining dimensions.
- Defining categories.

LOGICAL DESIGN PHASE

The logical design phase is the process of adding some refinement to the underlying data structure - a star schema

data model will be discussed in the following chapter. The goal of this phase is to provide complete, fully-defined logical data entities for the implementation of the physical design phase. The activity includes the following procedures[11]:

- Define and characterize entities: measure, dimension, and category detail.
- Define entity attributes and their relationships.
- Translate the information package diagram into a star schema.

PHYSICAL DESIGN PHASE

In this stage, the Decision Making Database System is moving into the process of implementing the data mart model in a practical manner. Visual Basic (version 4.0) enterprise edition is the chosen coding language for the design. Access 7.0 for Windows 95 is the sole database support for the system's back-end. The activity of this phase involves the following general steps:

- Creating the back-end supporting database system, designing and applying Access system built-in action queries to construct the tables, and stuffing the database tables with the related data.
- Building the new forms and modifying their properties by picking up and dropping all applicable customer controls

from the toolbox. This includes data control, a built-in control used as a gateway between the application forms and selected data sources. Such a control provides a tangible way of view to navigate the database system with little code required.

- Writing and attaching customer codes to the target object's event, in response to user- or system-generated occurrences.
- Designing fact-filling reports is one of the distinguishable features of the data mart model. The Crystal Reports control is one of the powerful additions to the Visual Basic Integrated Development Environment(IDE), which is used heavily in this phase.

SYSTEM VALIDATION

After the Decision Making Database System (DMDS) design completely goes through the previous steps - from the abstract conceptual model phase to the physical implementation phase - the system validation is the ultimate and inevitable process to finalize these preliminary development stages.

The function of the system validation is to review specification, design, and coding to assure that all performance requirements are being met. Later in this report, three interrelated performance tests will be

discussed in sequence: unit testing, integration testing and system testing.

The final goals of performance tests are to systematically uncover possible errors and correct them in a prompt manner. This is done in order to provide a satisfactory guarantee of the DMDS software's quality.

The above descriptions offer a glance at three aspects of the project approach. Later in this report, each of these phases will be discussed in detail and their specific objectives and expectations will be addressed.

CHAPTER FOUR

CONCEPTUAL MODEL DESIGN PHASE[11]

INFORMATION PACKAGING

By nature the present businesses' (including retail business') data are multitudinous. That is, the input data, which the data mart system is working with, are multidimensional. For example, if one picks up any promotional sale item in a supermarket, say, *Miller High Life Beer,* the gathering data related to this item, usually referred to as *information packaging*, can be dissected into the following segments:

- **Product**: *Miller High Life Beer.*
- **Promotional Period**: November 5, 1997 till November 11, 1997.
- **Location:** Store A, San Bernardino, California, USA.

As listed above, this sale product is associated with at least three data dimensions in order to properly express itself. Of course, there exist other key measurements besides *Product*, *Promotional Period*, and *Location*. They can describe the properties of the product (such as **Promotional**

Price: $4.99, or **Package**: 12/12-oz cans, etc.)in a multidimensional way.

INFORMATION PACKAGING METHODOLOGY

By the same token, there are a handful of design techniques and methodologies in the market today that are devoted to delivering data mart strategies and solutions.[12] However, information packaging methodology is the choice for this project. This methodology starts with collecting requirements from the requester, then analyzing the specification of these requirements; designing the prototype; implementing and testing; and finally delivering the product to the customer. If the requester is not satisfied, then the above procedure will be repeated.

However, the uniqueness of the information packaging methodology is in using the information package diagram to clearly define the scope of user requirements, and adapting the star schema technique in implementing the data mart paradigm. These two distinctive innovations are the essence of the conceptual model design and the logical model design, respectively.

INFORMATION PACKAGE DIAGRAM

Due to the ambiguity and the uncertainty of business query requirements from the data mart users, it is difficult

for the project developing team to capture/summarize precise information as the basis of overall design.

Therefore, an information package diagram is an effective tool to address this difficulty. This diagram simplifies the collecting key business requirements process and decides how multidimensional data should be presented to the data mart user. The information package diagram has three main components: dimensions, categories, and measures/facts.

DEFINING DIMENSIONS

As previously stated, the information regarding a sale item is multitudinous. The dimension is one of the standard indexes for the user to access the data, so is the system using dimensions to define and to present the acknowledgment information to the data mart user. Consider a dimensional example that relates to the project:

- **The Product dimension** includes the product category, size and packaging, product ID, unit price, units in stock, units on order and reorder level.
- **The Time dimension** contains promotion starting and ending dates, the transaction day of the year, the transaction week of the year, etc.

- **The Customer dimension** describes the customer's ID and personal information, the bank name of bounced check(s) and account information.

However, there is one function related to dimension - aggregation - which the data mart user can greatly appreciate. Aggregation refers to the summarized data in one- or multi- dimensional fashion. For instance, a headquarters' executive can use the aggregation function to see stock items across the branch stores or sale figures stretched from a daily to a yearly basis.

DEFINING CATEGORIES

Next, category is a series of classification levels within a dimensional hierarchy. The detail category is the smallest unit level within a dimension. The user of data mart can use drill up or drill down techniques to scale up or scale down one or more categories within a dimension in order to formulate a valid data point.

DEFINING MEASURES/FACTS

In the bottom section of the information package diagram is the component of measures/facts. A measure is an indication of dimensions in terms of quantities, money, or other measuring units. For instance, the daily sale of store #A in dollars is one of the measures.

SAMPLE OF CONCEPTUAL MODEL

In order to solidify the usage of the information package diagram in the data mart conceptual model design, the following is one of the requirements of the project, which the data mart needs to address:

Users of the data mart need to see bad customers detailed by check information, including check number, issuing bank, bank account number, and notice-sent dates over the last three years. The bad customers list should be presented with customer ID, personal information(address, etc.) and the branch store which received the bad check(s).

When analyzing the information to design the data mart, information must be broken down into related segments as follows:

(1) **Need to see bad customers .. bad customers list should be presented with customer ID, personal information(address, etc.) and the branch store which received the bad check(s).**

Dimensions: bad customers list.

Measures/Facts: customer ID, personal information, and branch store, which received the bad check(s).

(2) **detailed by check information, including check number, issuing bank, bank account number, and notice-sent dates ..**

Dimensions: check number, issuing bank and bank account number, and as well as notice-sent dates.

(3) ..over the last three years.

Dimensions: time periods of three years.

Figure 4.1 Information Package Diagram

Information Package: Bad Customers Analysis

Dimensions →

ALL Time Period (count)	ALL Bad Customers (count)	ALL Bad Check Numbers (count)	ALL Issuing Banks (count)	ALL Bank Account Numbers (count)	ALL Notices Sent Dates (count)		
Year 3	Bad Customers 100	Checks 3	Banks 19	Bank Accounts 102	Notices 3		
1st Year 1		1st Check 100	Bank of America 39		1st Notice 100		
2nd Year 1		2nd Check 43	Bank of Redlands 12		2nd Notice 29		
3rd Year 1		3rd Check 22	...		3rd Notice 13		
Day 365							

Categories ↓

Measures/Facts:

Customer ID, Personal Information, Branch Store which received the bad check(s).

CONSIDERATION IN INFORMATION PACKAGE DIAGRAM

When the time comes to implement the conceptual design for the data mart architecture, there is one rule of thumb one should take into consideration. That is, do not try to cover all possible requested information in one information package diagram, i.e., overload the dimension within the diagram. Try to create other diagrams as needed and minimize the number of dimensions to under 8, and the number of categories to under 6 per diagram. The shaded area on the information package diagram in Figure 4.1 reminds users of this guideline. The essence of the data mart is its smaller design, which makes it better and easier to implement. Consequently, the goal for the data mart users - providing a higher degree of users' satisfaction - is completely achievable.

Figure 4.1 is also a sample presentation of a completed information package diagram extracted from the project. Within this information package diagram, the attention should focus on each cell under the high-level dimension columns. These cells are all filled with the title that is associated with the categories(or sub-categories), and the value within each cell represents the unique occurrences of this category/sub-category.

CHAPTER FIVE

LOGICAL DESIGN PHASE[11]

After the information package diagram is constructed, the project can proceed to the second level of development - building a logical data model for the data mart. The finished information package diagram provides the conceptual foundation for the smooth deployment of star schema data modeling.

MISSION OF THE DATA MART SYSTEM

Prior to initiating the logical design of the star schema, let us review the mission statement of the data mart system. A data mart system enables the user to use queries heavily and analyze data thoroughly across many facets of information resources. A perfect data mart system allows the user to have comprehensive insight as well as full control over the data. Such control includes not only extraction, aggregation, and summarization of the needed data, but also screening out of unwanted data.

STAR SCHEMA

The star schema is a query-based optimizer. Unlike other traditional database modeling techniques, the modeling

of star schema organizes data entities in a way that reflects the decision-makers' view of their importance and highlights the ease of navigating/commanding operational aspects of data.

LOGICAL ENTITIES

When converting the information package diagram into star schema, three types of logical entities within the data mart structure have been defined: dimensions, measures, and category details. The number of dimension entities is different from the number of measure entities, in that dimension entities will not increase during the time of operation.

Within a star schema model, the dimension entities are represented as the tip-points of the star. In contrast, the data contained within the measure entities grow very large over time. As a result, the measure entity is the primary focus point of the user's query operations. Hence, it is located at the center position of the star schema diagram. The category details entity, on the other hand, is at the lowest level of information detail; it provides a well-informed qualitative content to assist the user in the decision-making process.

TRANSLATION FROM CONCEPTUAL TO LOGICAL MODEL

However, in order to transfer smoothly from the information package diagram into the star schema, the measure entities and dimension entities of the conceptual model need to be respectively redefined into their counterpart entities in the logical model.

LOGICAL MEASURE ENTITY

A logical measure entity is composed of the lowest-level category within each dimension, plus each measure entity of the information package diagram, as illustrated by TABLE 5.1.1.

TABLE 5.1.1 Measure entity translation from the information package diagram of Figure 4.1.

Bad Customers Analysis Measure
Day
Bad Customers
3rd Checks
Bank of Redlands
Bank Accounts
3rd Notices
Customer ID
Personal Information
Branch Store, which Received the bad

LOGICAL DIMENSION ENTITY

On the other hand, the logical dimension entity is the conversion of all categories under each dimension of an

information package diagram. This is seen as columns on the following logical tables. See TABLE 5.1.2 to TABLE 5.1.7 for an example of this discussion.

TABLE 5.1.2 Time period dimension entity translation from the information package diagram of Figure 4.1.

Time Period
Years
1st Year
2nd Year
3rd Year
Day

TABLE 5.1.3 Bad Customers dimension entity translation from the information package diagram of Figure 4.1.

Bad Customers
Bad Customers

TABLE 5.1.4 Bad Check Numbers dimension entity translation from the information package diagram of Figure 4.1.

Bad Check Numbers
Checks
1st Check
2nd Check
3rd Check

TABLE 5.1.5 Bad Check Issuing Banks dimension entity translation from the information package diagram of Figure 4.1. (The dots represent too many banks in this category)

Issuing Banks
Banks
Bank of America
Bank of Redlands
..

TABLE 5.1.6 Bank Account Numbers dimension entity translation from the information package diagram of Figure 4.1.

Bank Account
Bank Accounts

TABLE 5.1.7 Notice-Sent Dates dimension entity translation from the information package diagram of Figure 4.1.

Notice-Sent Dates
Notices
1st Notice
2nd Notice
3rd Notice

ENTITIES RELATIONSHIP

The relationship between the logical entities of measure and dimension is a typical one-to-many; this can be easily verified, as shown in Figure 5.1(illustrated in the next page). One measure entity instance relates to many dimension entity instances in the star schema diagram.

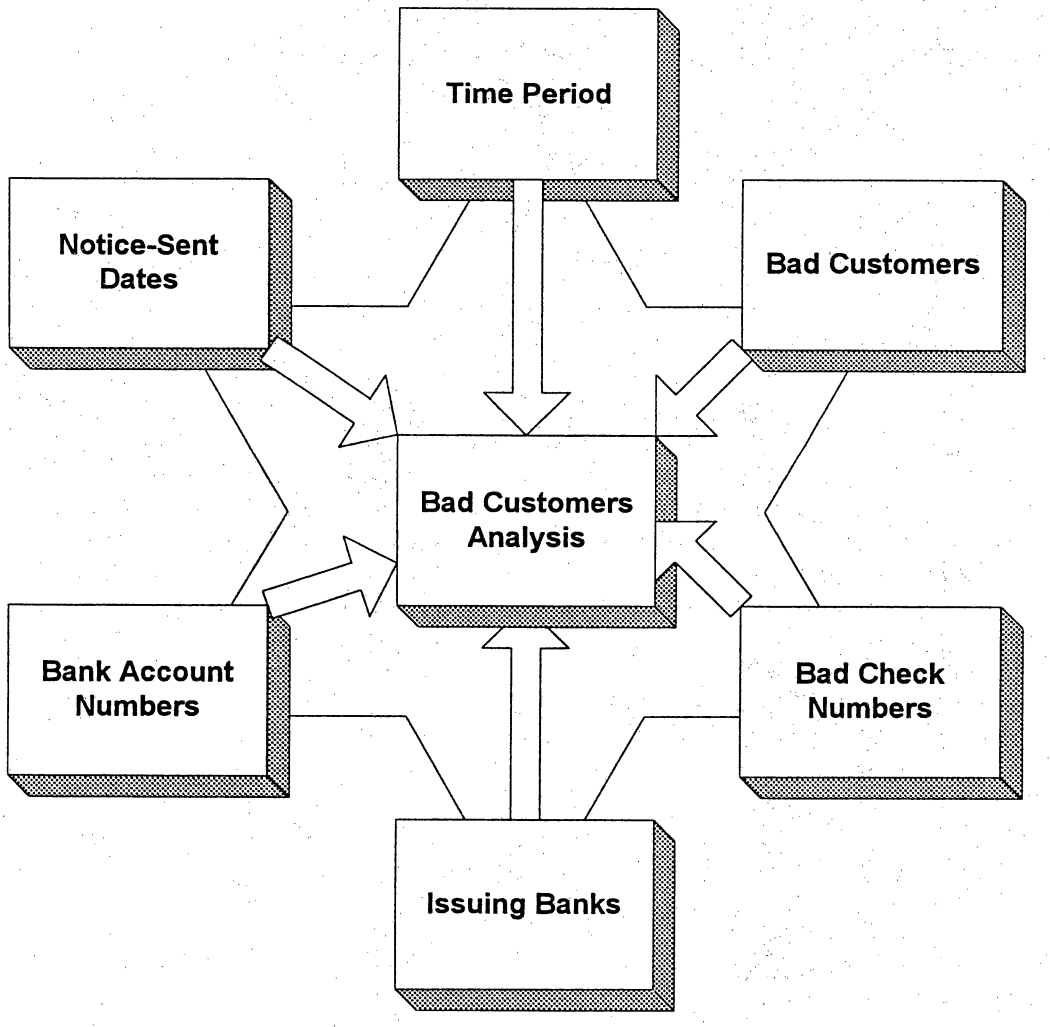


Figure 5.1 Star Schema relationships between dimension and measure entities

CHAPTER SIX

PHYSICAL DESIGN PHASE

So far, two different levels of model design have been presented during the development process of the data mart, i.e., the completeness of conceptual and logical model designs. The next refinement stages will focus on the physical model design and its implementation processes.[11] Subsequently, by concluding all of the stages of information packaging methodology, the final product - Decision Making Database System (DMDS) will be built and delivered in a manner more closely matched with what users have requested.

In this chapter, the transition will be made from the logical star schema to the building of the physical design model. [11]The three types of physical entities will be addressed: category detail, measure, and dimension. The relationships among these entities will also be illustrated by the example used throughout the previous chapters.

PHYSICAL ENTITIES

As stated in the previous sections, the measure entities are made up from the lowest level of the detail category (last cell) within each dimension and all the measure entities in the measures/facts section of information package diagrams[11].

[11]On the contrary, the category detail entity contains the data that are either qualitative or descriptive in nature to provide the information for the decision-makers. Usually, category detail entities function as a valuable reference, and these entities are translated from the logical design entities into the physical design tables.

SAMPLE OF PHYSICAL DESIGN TABLES

The following physical database tables are the category detail entities for this project.

TABLE 6.1 Store Profile

Field Name	Data Type	Description
StoreNumber	Text(50)	Assigned Store Number
StoreSize	Text(15)	In Square Feet
GeneralManager	Text(50)	Last Name, First Name
TotalEmployees	Number(Long Integer)	Total Number of Employees
YearSale	Currency	Store Yearly Sale
WeekSale	Currency	Store average Weekly Sale
CustPopulation	Text(50)	Store Approx. Customer Population
CustIncome	Text(50)	Store Approx. Customer Income Level
Address	Text(255)	Store Street Address
City	Text(50)	Store's City
PostalCode	Text(5)	Store's 5-digit Zip Code
StateOrProvince	Text(2)	Store's State Abbreviation.
PhoneNumber	Text(30)	Store's Main Phone Number
ModifyBy	Text(50)	User's Name
ModifyDate	Date/Time	Modified Date in MM/DD/YY Format

TABLE 6.2 Categories

Field Name	Data Type	Description
CategoryID	AutoNumber(Long Integer)	Number Automatically Assigned to a New Category.
CategoryName	Text(15)	Name of Food Category
Description	Memo	A text of note
Picture	OLE Object	A Picture Representing the Food Category

TABLE 6.3 Customers

Field Name	Data Type	Description
CustomerID	Text(8)	Customer's Driver License Number
Name	Text(30)	Customer's Name: First name, Last name
Address	Text(255)	Street Address
City	Text(50)	Customer's City
State	Text(2)	Customer's State Abbreviation
ZIP	Text(5)	Customer's 5-digit ZIP Code
PhoneNumber	Text(30)	Customer's Resident Phone Number
Notes	Memo	A text of note
CK1	Text(50)	Bad Check #1
CK2	Text(50)	Bad Check #2
CK3	Text(50)	Bad Check #3
CKDate1	Text(50)	First Bad Check Notice Date
CKDate2	Text(50)	Second Bad Check Notice Date
CKDate3	Text(50)	Third Bad Check Notice Date
CKNT1	Yes/No	Was the First Bad Check Notice Sent Out?
CKNT2	Yes/No	Was the Second Bad Check Notice Sent Out?
CKNT3	Yes/No	Was the Third Bad Check Notice Sent Out?
BankName	Text(50)	Name of Bank That Issued First Bad Check(Assumed one Bank/per customer)
AccountNo	Text(50)	Customer's Bad Check Account
CKF1	Text(50)	Branch Which Received the First Bad Check
CKF2	Text(50)	Branch Which Received the Second Bad Check
CKF3	Text(50)	Branch Which Received the Third Bad Check

TABLE 6.4 Products

Field Name	Data Type	Description
ProductID	AutoNumber(Long Integer)	Number Automatically Assigned to New Product.
ProductName	Text(40)	Product Name
SupplierID	Number(Long Integer)	Same Entry as in Supplier's Table
CategoryID	Number(Long Integer)	Same Entry as in Category Table
QuantityPerUnit	Text(20)	(e.g., 24-Count Case, 2-Liter Bottle)
UnitPrice	Currency	Product Unit Price
UnitsInStock	Number(Integer)	Product Units in Stock
UnitsOnOrder	Number(Integer)	Product Units on Order
ReOrderLevel	Number(Integer)	Minimum Units to Maintain in Stock
Discontinued	Yes/No	Yes Means Item Is No Longer Available
UnitsSt1	Number(Long Integer)	Store #1 Product on Stock
UnitsSt2	Number(Long Integer)	Store #2 Product on Stock
UnitsSt3	Number(Long Integer)	Store #3 Product on Stock
UnitsSt4	Number(Long Integer)	Store #4 Product on Stock
SaleSt1	Currency	Store #1 Product Sale Price
SaleSt2	Currency	Store #2 Product Sale Price
SaleSt3	Currency	Store #3 Product Sale Price
SaleSt4	Currency	Store #4 Product Sale Price
SalePrice	Currency	Product Default Sale Price
SaleStart	Text(50)	Sale Start Date
SaleEnd	Text(50)	Sale End Date

TABLE 6.5 Supplier

Field Name	Data Type	Description
SupplierID	AutoNumber	Number Automatically Assigned to New Supplier
CompanyName	Text(40)	Supplier Company Name
ContactName	Text(30)	Contact Person Name
ContactTitle	Text(30)	Contact Person Title
Address	Text(60)	Street Address/ P.O. Box
City	Text(15)	Supplier's City
Region	Text(15)	State or Province
PostalCode	Text(10)	Postal Code
Country	Text(15)	Country
Phone	Text(24)	Phone Number and Area Code
Fax	Text(24)	Fax Number and Area Code

TABLE 6.6 Daily Sale

Field Name	Data Type	Description
StoreNumber	Text(50)	Assigned Store Number
TodaySale	Currency	Today's Transaction Amount
PreviousSale	Currency	Previous Day's Transaction Amount

TABLE 6.7 Sale Data by Region

Field Name	Data Type	Description
StoreNumber	Text(50)	Assigned Store Number
Year	Number(Long Integer)	Year of Transaction
Month	Number(Long Integer)	Month of Transaction
Amount	Currency	Transaction Amount
Region	Text(10)	(e.g. West, South, North etc.)

Unlike the category detail entity, the dimension entity of the physical phase may not be translated directly from the logical entity into the physical database table. Usually, the dimension entity works as one of the available pick-up items for the list boxes or drop down menus in the system's front-end.[11]

ENTITIES RELATIONSHIP

Before the actual implementation of the Decision Making Database System (DMDS) from a star schema into a physical model, it is vital to identify the relationship between each measure entity and its associated dimension entities[11]. These relationships are essential for the success of the physical design phase and the implementation of the project later on. A data mart physical model represents this relationship, as shown in Figure 6.1. (illustrated in the next page)

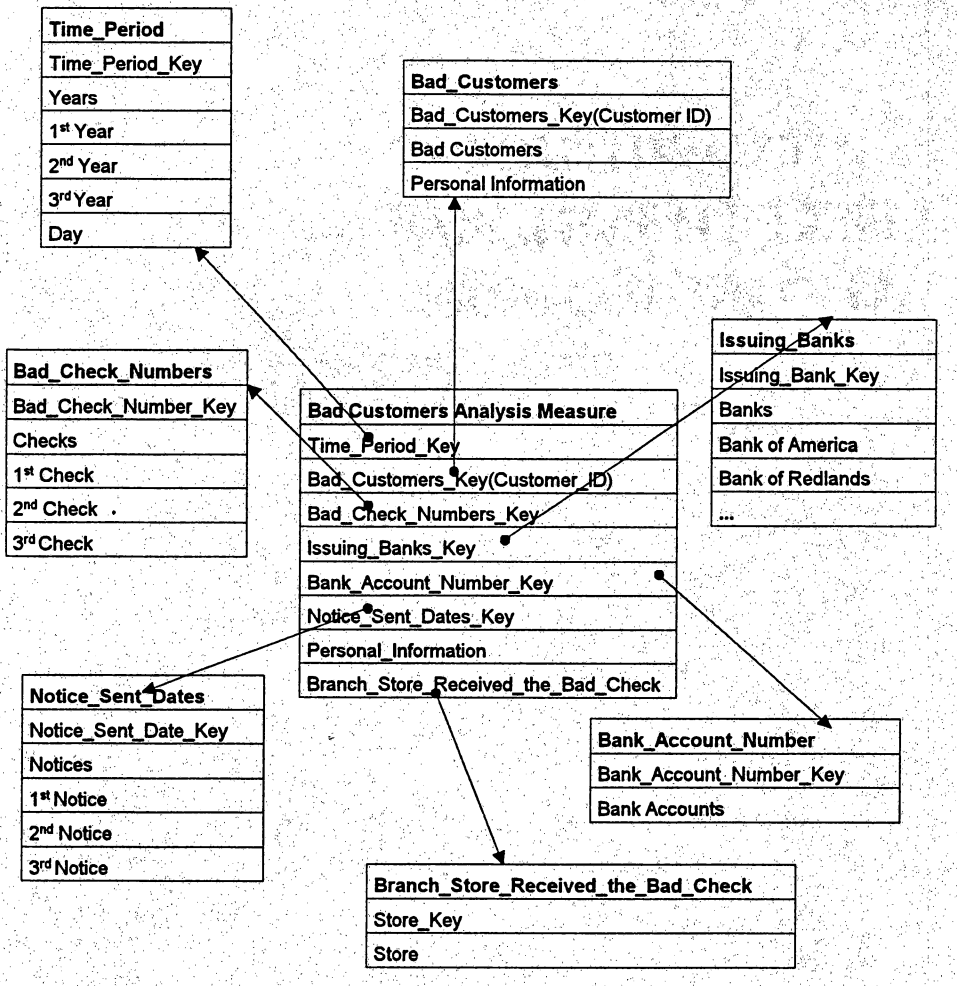


Figure 6.1 Physical Data Model relationships between entities.

CHAPTER SEVEN

PROJECT IMPLEMENTATION

The project implementation is the embodiment of the application design using Visual Basic. In this chapter, some general processes and some features of the DMDS will be highlighted. The processes include creating and managing a database, developing fully functional screens of input forms, object linking and embedding (OLE), generating information-filled reports, and lastly, providing application-level security.

CREATING AND MANAGING A DATABASE

The database tables of this project (i.e. *Project.mdb*) were either created or modified by using the built-in action query of Access. Access was chosen for this design mainly because 32-bit Visual Basic 4 shares its database engine (Jet version 3.0) with the 32-bit version of Access 7. Consequently, the Jet 3.0 data format can be read and easily deployed in an environment that contains both the 32-bit versions of Visual Basic and Access software.[9, 10]

The relationships of physical design entities that developed from the previous chapters are converted to the database table format. However, these tables for the

implementation phase are derived from the sample database, *Northwind.mdb*, which came with the Access software package and is generally on the directory tree (\MSOffice\ACCESS\SAMPLE\Northwind.mdb).

The Decision Making Database System (DMDS) application has 23 tables and 33 queries total. The action queries of Access were used to create, append, update, and delete operations to manipulate the table and associated data within the database.

The programming code underlying the action queries is Structured Query Language (SQL), the language of the database which Visual Basic and Access can share with in order to retrieve the match-specific criteria data.[13]

Moreover, SQL provides a wide variety of capabilities and functionality. It can be used: [14, 15]

- To interactively retrieve and display data.
- To work with programming languages. SQL statements embedded in code provide access to the relational database.
- To manage database structures. SQL allows one to create and modify the structure.
- To control access to databases through the granting and revoking of privileges.

An example of table SBDCustomers, which is created on the *Project.mdb* using the Microsoft Access SQL action query, is listed below:

```
SELECT DISTINCTROW Customers.CustomerID,
Customers.Name, Customers.Address, Customers.City,
Customers.State, Customers.ZIP, Customers.PhoneNumber,
Customers.Notes, Customers.CK1, Customers.CK2,
Customers.CK3, Customers.CKDate1, Customers.CKDate2,
Customers.CKDate3, Customers.CKNT1, Customers.CKNT2,
Customers.CKNT3, Customers. [Bank Name],
Customers.AccountNo, Customers.CKF1, Customers.CKF2,
Customers.CKF3 INTO tabSBDcustomers FROM Customers WHERE
(((Customers.CKF1)="SBD")) OR (((Customers.CKF2)="SBD")) OR
(((Customers.CKF3)="SBD"));
```

By using SQL statements as the database query language in the code of the project, Visual Basic can communicate with Access as well as with other computer environments to retrieve, filter, and sort the needed database information on the fly.[13]

Like any language, SQL has its own dialects. The example in the previous paragraphs is written in terms of Access SQL. Access SQL, as supported by the Microsoft Jet engine, varies slightly from ANSI standard SQL-92. In particular, it does not support subqueries, data control language or data definition language statements[14]. The detail of SQL syntax in Access (and Visual Basic) is beyond the project's scope of discussion, which this document does not intend to address. But there is no doubt that having a solid grounding in SQL programming is a must for this

project and developing commercial database applications later on.

DEVELOPING FULLY FUNCTIONAL SCREENS

In this section, the Visual Basic Graphic User Interface (GUI) techniques and data-bound controls are intensively used to create the gateway to access the underlying database for the decision making purpose.

The bound data control has two main functions in this project. One is to connect the front-end of the application for the user business queries input, while the other is to display the feedback of requested information from the back-end supporting database, *project.mdb*. Throughout the design of the Decision Making Database System (DMDS), the use of data control greatly simplifies the coding task and provides an excellent visual navigating tool for the user.

Since the data control needs to be incorporated closely with input and output bound controls of Visual Basic, it is critical that these bound controls are well understood before adding them into the project's application.

The bound controls that are to be used with the data control in this project include the following: [15]

- **Check Box** Used to provide read/write access to a specified Boolean or bit data field.

- **Label**: Used to provide display-only access to a specified text data field.
- **Text Box**: Used to provide read/write access to a specified text data field.
- **Picture Box**: Used to display a graphical image from a bitmap, icon, or metafile on the form. As a bound control, it can be used to provide read/write access to a specified image/binary data field from one of the application tables.
- **List Box**: Used with the *AddItem* method. When bound, it may be used to provide read/write access to a specified text data field selected from the list.
- **Data-bound combo box**: Used to create a bound combination list box and text box, or a simple drop-down list. The list can be filled automatically from data control.

To use the bound controls in conjunction with data control, one or more data controls first need to be placed on the form. Then, draw the desired bound control on the same form as the data control. The data control establishes a link between the back-end database and the front-end

target input/output bound controls by doing the following special procedures to the bound controls: [15]

1. Set the *DataSource* property to specify the data control to which it will be bound.
2. Set the *DataField* property to a valid field in the data control's record set.

However, when running the application, any changes in the data control record set, such as modify record are automatically updated in the associated field of the underlying database, with little or no coding involved.

OBJECT LINKING AND EMBEDDING (OLE)

With the rapid growth of automated technologies, the information system for the corporate knowledge base seldom comes into unified standard format. The fact is that every corporation may use several different software/hardware vendors simultaneously to manage the corporation's knowledge base asset.[11] The obsolete nature of the one-tool-fits-all strategy and current trends to develop a supporting multiple data format technology have made OLE one of the highly desirable features for any commercial software.

In the information_browser module of the Decision Making Database System (DMDS), OLE is the underlying technology for this design. The growing importance of OLE for successfully transfer compound documents into corporate

knowledge asset (i.e. DMDS format) result from the increasing demand for compound documents formats (e.g., spreadsheets, bit maps, memos, or reports etc.), while are rapidly changed. That format change can occur either in the data mart or in the office automation communities. [11]

As the title of this section implies, in the object-oriented programming environment (such as Visual Basic), there are two ways to connect external data into the application; that is, either by linking or by embedding.

[15, 16]The principle behind the object embedded option is once the OLE object in the application is triggered, the entire object will be copied from the original file. As a result, all the tools on the menu of the original application associated with this object will come to life for users to view or modify.

On the other hand, the linked objects option is the one used throughout this project. [15, 16]The object linked option works by getting the application to hold the pointer and link it to the file containing the external data. When the user activates the OLE object, the interface form of Visual Basic will replace the original application associated with that type of external data.

For example, if one activates the OLE object by double clicking the EmployeeOfTheMonth bitmap image file on the Visual Basic form, a Microsoft Paint pallet with the

EmployeeOfTheMonth file will load into view for the user to edit.

GENERATING INFORMATION-FILLED REPORT

One of the major advantages for the company that implements the data mart paradigm is gaining important information to exclusive of others in the same field.[11] In the Decision Making Database System (DMDS), there is a module which provides a tool to assist the user in better understanding the current status of the company, from product inventory to the bad check customers list through the final visualization-report.

In the module of frmProductPricing and frmBadCheckCustomerTB, several reports can be generated based on the user's different query criteria. In the frmProductPricing module, the store product inventory reports will provide critical information for management level personnel to have smooth daily store operations. Some examples of such requests are:

- What product-items are currently below re-order level and re-order not completed?
- What product-items are the most recent order?
- What product-items are promotional sale items?

In the BadCheckCustomerTB module, the confidential reports list the store's bad customer information. These reports work as a useful pointer for the retail business and other businesses alike to meticulously watch the cash flow of transactions. Moreover, these reports also serve as updated documents for the communication records (such as dates notices were sent etc.) between the company and the customer who repeatedly issued the bad check. The following are the report titles of this category:

- Customers List- one bounced checks.
- Customers List- two bounced checks.
- Customers List- three bounced checks.

In order to generate the report through the Decision Making Database System (DMDS), the Visual Basic add-in--the Crystal Reports program and its customer control--are two indispensable elements for such an operation in response to the user's required output.

The Crystal Reports program is a substantial part of the Visual Basic Integrated Development Environment (IDE). Through the Crystal Reports program, the project developer designs the format of the report's appearance and contains the field information of the underlying database.

[15]Generally speaking, there are three sections in the Crystal Report design template: Page header, Page footer, and Details. The Details section is equivalent to the area

of record details in any regular business report. Moreover, the Details section is the place where the report designer can lodge field information of attached database tables. However, the Page header usually displays report titles, descriptions, and dates printed information. On the other hand, Page footer usually only shows the page number and is repeated at the bottom of every page in the report.

Building the report writing capability into the system is done by dropping the Crystal Reports customer control onto the form. Users can interact with this form at design time and set its control properties at run time to achieve this capability. The specific properties of the Crystal Reports customer control used in this project include: [15]

- **Destination** The report's destination: Window = 0, file = 1, printer = 2
- **ReportFileName** A string expression indicating the name of the report that the user wants to print in response to an application event.
- **WindowTitle** A string expression to be displayed in the report window's title bar.
- **WindowTop** Sets the Y position of window in pixels.
- **WindowLeft** Sets the X position of window in pixels.

- **WindowHeight** Sets the height of the report window in pixels.
- **WindowWidth** Sets the width of the report window in pixels.

The details of the code written to generate information-filled DMDS reports are shown in APPENDIX A. And the Complete report printouts are attached in APPENDIX B.

APPLICATION-LEVEL SECURITY

Most quality commercial applications in today's market have some kind of security mechanisms to protect valuable and restricted information and/or databases. The Decision Making Database System (DMDS) is no exception. In this section, two layers of security protection, namely the multi-user login/logout process and programmable access rights for restricted operations, will be the center of consideration.

Currently, most software programs that devote themselves to client/server multi-user environments should have some basic levels of security schemes. One such fundamental security scheme is to install a user login/logout process to limit user access to the system and its associated databases.

The login/logout process of this system needs the application administrator to create a list of valid users

and force the users to log into the application using authorized passwords. Typically, the login entry form of this process has a list of qualified users' names. The system login routine will check the user ID and password against the user profile on the system once the user completes and submits the required information. As usual, after three failed attempts to access the system, the user will be flushed out.

The second scheme of the project to safeguard the application valuables is programmable access rights for the restricted operations. In this scheme, the system security focuses not only on the authorized users who gain admittance to the system, but also on different users that may have distinct system privileges. For instance, the system administrator can limit some levels of users' accessibility to examine those highly confidential bad customers' reports or even to prevent invalid users to run mission-critical operations. Such operations include promotional price updates or any disastrous mistakes that may erase the transaction records of any branch stores, and so on.

Before implementing the access rights scheme into the DMDS, the scale of access rights is deployed as follows in Table 7.1.

TABLE 7.1 The access rights levels

Rights Level	Access Rights
Level 1	Read
Level 2	Edit, Update
Level 3	Add, Delete
Level 4	Report

In Table 7.1, each successive level adds additional privileges to the previous level. Level 4, for example, can have all six defined rights to manipulate the system. Moreover, the sample users' rights table (Table 7.2) needs to be set up for each authorized user before this feature can be put into practice. Any users who attempt access to operations of any kind must have their own access profile defined for that action on the system log. If the user violates this rule, the warning will be given and the system will terminate immediately.

TABLE 7.2 Users' Rights Table

User ID	User Level
Bill Clinton	Level 2
John Doe	Level 1
Paula Jones	Level 2
Donald Trump	Level 3
Jerry Maguire	Level 3
Jim Carrey	Level 4

CHAPTER EIGHT

SYSTEM VALIDATION

The system validation is the process of critical tests being conducted and the results being evaluated against original design specifications and intended functionality (validation requirements). The purpose of system validation is having assurance about Decision Making Database System (DMDS) software quality. This guarantees system performance reliability.

Later in this chapter, a series of different test schemes are discussed in sequence. However, each test scheme has its own objectives that the tests intend to uncover. In the sections that follow, each test scheme is presented in detail.

UNIT TESTING

[17]After the program code is developed, executed, reviewed, and inspected, the unit test is the initial step in the software testing process. The unit test focuses on the smallest functional unit of design. This smallest functional unit can be single programs, internal procedures, or independent modules in an isolated test environment. In the Decision Making Database System (DMDS) the unit test usually includes some of the following items to check for in

each functional unit:Ensure that all the command buttons work as expected.

- Check normal and abnormal program termination.
- Verify the handling of all valid input data types.
- Check data retrieval times (i.e. how long it takes to retrieve data?).
- Verify the handling of error conditions.

While this is an enormous workload, unit testing generally yields the highest number of detected problems within all testing techniques.[11, 17] A loose and sloppy unit test will cost someone a great deal of time in later stages of integration testing, system testing, and use. The results of the unit test for the Decision Making Database System (DMDS) are listed in Table 8.1(*Continued from* page 59 - 62)

Table 8.1 Unit Test Results

Forms	Tests Performed	Results
frmAdd	• verify the handling of all valid data types. • check the information box appearance. • Ensure Add command button works as expected.	***Pass***
frmBadCheck	• Check data retrieval time and accuracy. • verify normal and abnormal program termination.	***Pass***
frmBadCheckTB	• check the retrieving data accuracy through query operations.	***Pass***

	• test the report's printing dependability.	
frmClose	• check the Delete confirmation box appearance. • ensure that the Delete button works as expected.	***Pass***
frmCustomerUpdate	• test the consistency of data updated in local and headquarters' databases.	***Pass***
frmDailyNew	• check the accuracy of retrieving dates data.	***Pass***
frmDailySale	• test the timing of the pop-up screen after double-clicking the table.	***Pass***
frmDailyUpdate	• verify the accuracy of the Week-to-date and Year-to-date amount. • ensure the three command buttons work as expected.	***Pass***
frmEmployee	• test for proper retrieval and store related data. • verify the validation of the update operation.	***Pass***
frmEnd	• check the appearance of the form.	***Pass***
frmGraph	• review the correctness of graph representation. • examine the consistency of items within the drop-down menu.	***Pass***
frmInformationBrows-er	• analyze the connections among the directory boxes, the control box, and those OLE items. • check normal program termination.	***Pass***
frmLiveUpdate	• check that the simulation status bar agrees with its associated subtitles.	***Pass***

frmModifyInfo2	• check the information box appearance. • verify the functions of special keys - Edit and Update. • test the response time of the pop-up menu due to a change in the total number of employees.	***Pass***
frmOverView	• ensure the three command buttons work as expected.	***Pass***
frmPrivilege	• verify that access privileges are clearly defined for users.	***Pass***
frmProductMovement	• check the accuracy of retrieving product information. • Test the accurate assignment of newly-arrived products.	***Pass***
frmProductPricing	• check the information box appearance. • check the accuracy of retrieving product information. • test price-update work according to the pricing margin(%). • verify the correctness of inventory reports.	***Pass***
frmSelectOperation	• Check the correct links of selected items. • ensure the four command buttons work as expected.	***Pass***
frmSplash2	• check the form's appearance.	***Pass***
frmStoreProfile	• check the accuracy of retrieving store profile information. • examine consistency of items within the drop-down menu. • check the functions of	***Pass***

	the four navigation command keys.	
frmStoreProTable	• check the accuracy of retrieving store profile information.	***Pass***
frmStorMaint	• Check the correct links of selected items. • ensure the four command buttons work as expected.	***Pass***
frmSystemSecurity	• check users' ID against users' authorized passwords. • test the execution of the *three-strike-out* strategy.	***Pass***
Reports		
BelowReorderList, OnOrderItems, OnSale ReorderIncomplete,	• check the correct format setting. • verify the accuracy of the product information.	***Pass***
AllCustomers, Once, Twice, Three	• check the correct format setting. • verify the accuracy of the product information.	***Pass***

INTEGRATION TESTING

In essence, in the design phases of the Decision Making Database System(DMDS), it is necessary to repeatedly break down the system into manageable units for the sake of clarity. However, in the implementation phase, the interface among components, and how to build and integrate these units into a whole, need to have full consideration.

Although the underlying system components successfully went through unit testing, unfortunately, there is no

guarantee to deliver the system functionality defined in the design specification after the components are put together as one piece. Therefore, integration testing is a systematic technique for assembling software while at the same time conducting tests to uncover errors associated with interfacing.[17]

The integration tests in this system take individual units like form and verify their interfaces and link with other units of the overall DMDS and subsystems. The Table 8.2 is the summary of Integration Test results of this system.

Table 8.2 Integration Test Results

Components Integrated	Test Performed	Results
the DMDS Forms and Reports	• test the proper linkage between all forms and Crystal Reports functionality.	***Pass***
the DMDS Forms and their associated OLE applications	• test the proper linkage/embedding between OLE applications and the DMDS forms.	***Pass***

SYSTEM TESTING

After the Decision Making Database System (DMDS) software is completely tested, and meets all functional and

performance requirements, the final step in system validation process is called the system test. Since the installed DMDS application is only part of the underlying computer system, it is essential to have a system test to verify that all system elements (e.g., DMDS, hardware and other possible different operating system environments) team up to work properly.

[17]Classic system tests often include an acceptance test and a usability test. An acceptance test is conducted by the user of DMDS. Its purpose is to validate the system's functionality to match the users' expectation. That is, [11]the users feel ease at installation of the software and operate in a productive way in a short period of time.

The usability test evaluates the usefulness of system information modules which DMDS software provides in the views of the user. Therefore, the usability test can work as an effective tool for providing feedback to the software developing team to provide better service to the user in the next revision of the software.

CHAPTER NINE

USER'S MANUAL

This manual provides information about the Decision Making Database System(DMDS). It is written for management-level users who may or may not have much computer experience but are familiar with Microsoft Windows 95 or Windows NT operating systems. It contains information about retrieval, storing related data, navigating the customer or product profiles, and providing management with techniques to operate corporations' databases. The decision tree of the DMDS is illustrated form Figure 9.1 to Figure 9.7 of following pages.

Throughout this manual, the following conventions (TABLE 9.1) have special meaning:

TABLE 9.1 Conventions Table

Convention	Purpose
Access	Shows the user the path to get to this screen (form).
Notes	Shows the user what operating options are available on this screen, and describes the action executed for the chosen option.
Bold	Represents the command names on the screen, such as the **Finish** command key.
italic	Represents the DMDS form's name.

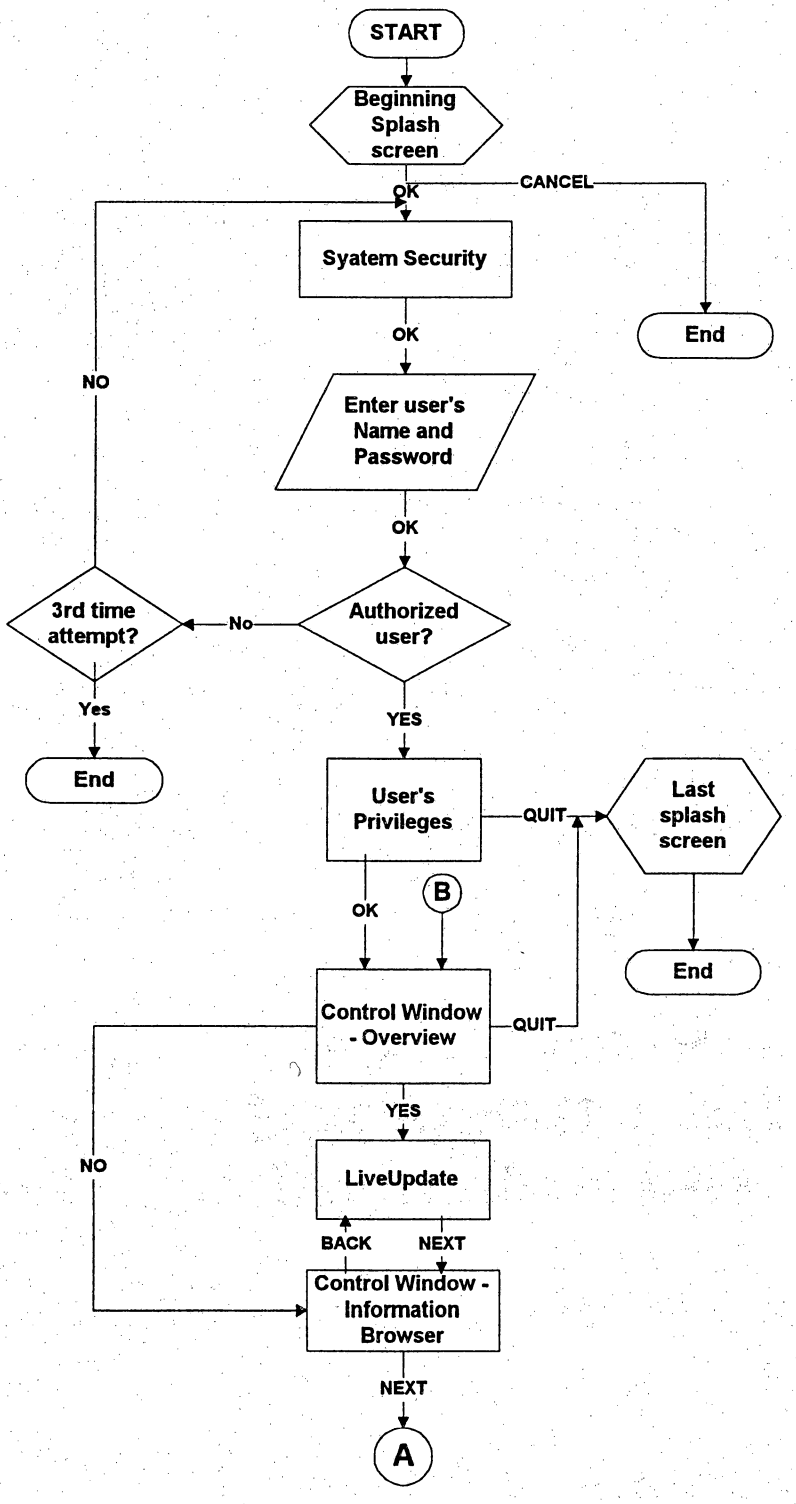


Figure 9.1 The Decision Tree of the Decision Making Database System(DMDS)

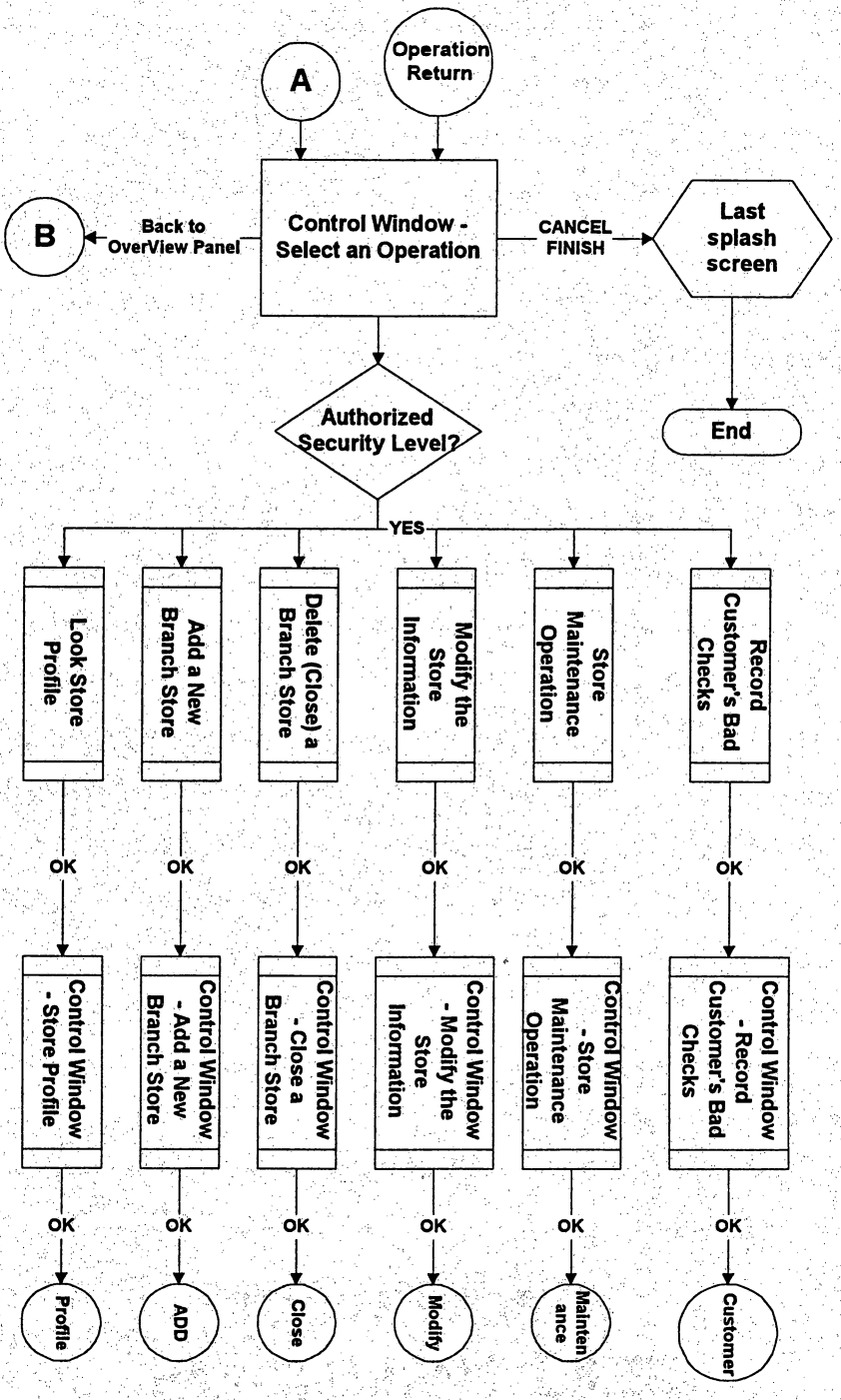


Figure 9.2 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

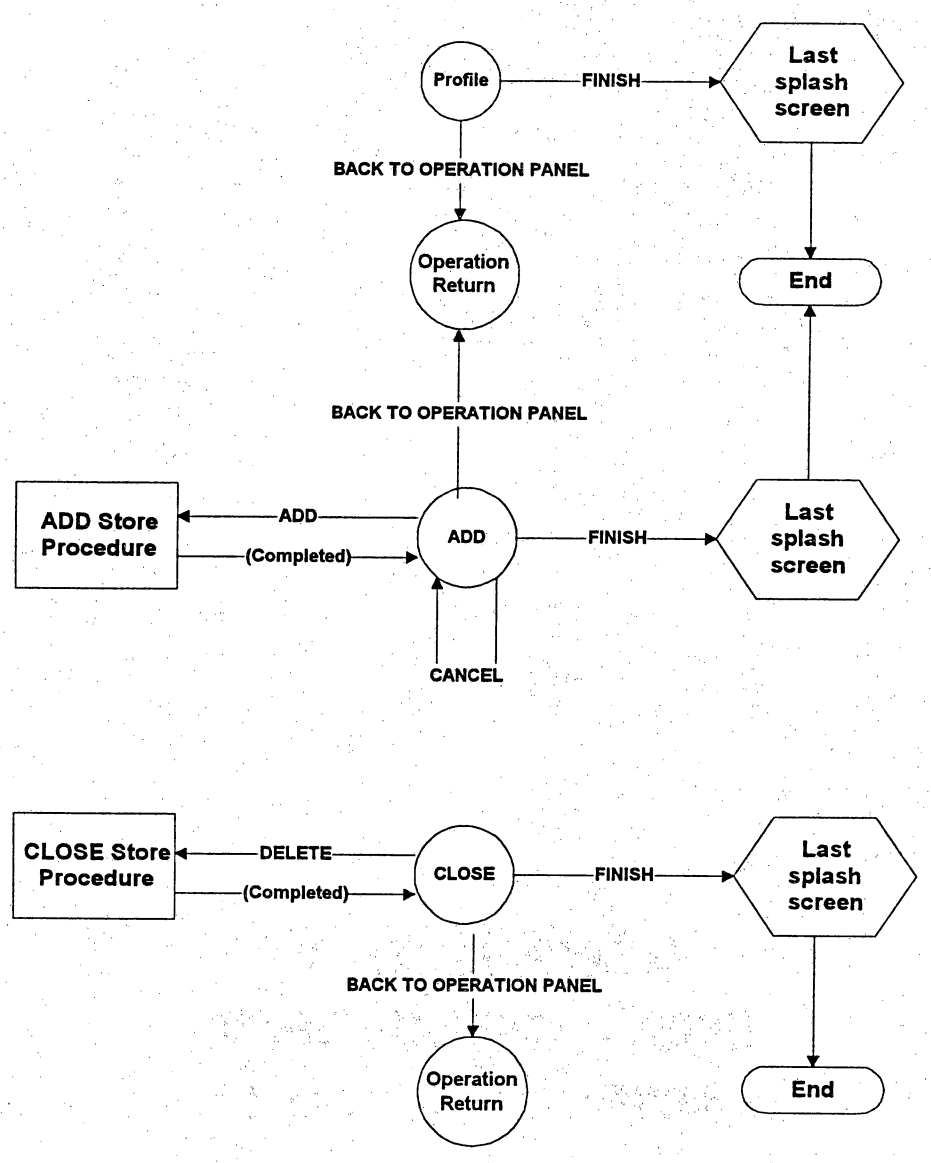


Figure 9.3 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

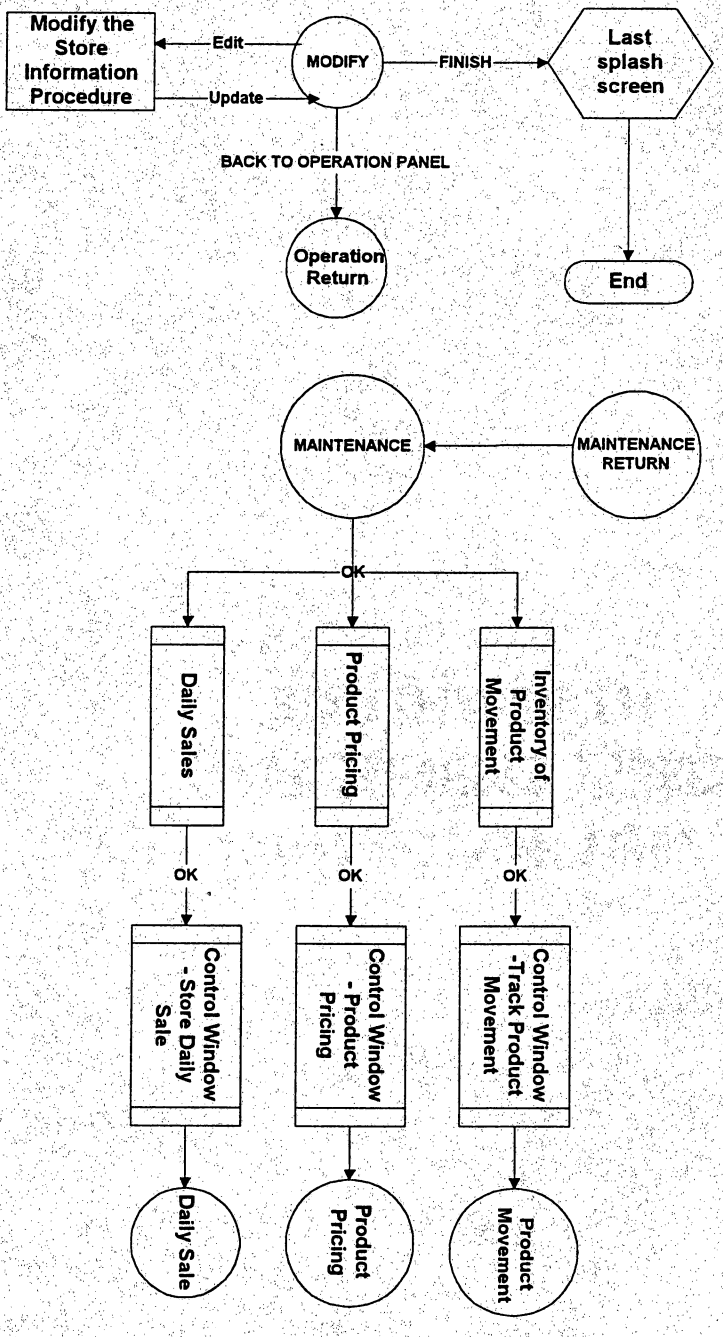


Figure 9.4 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

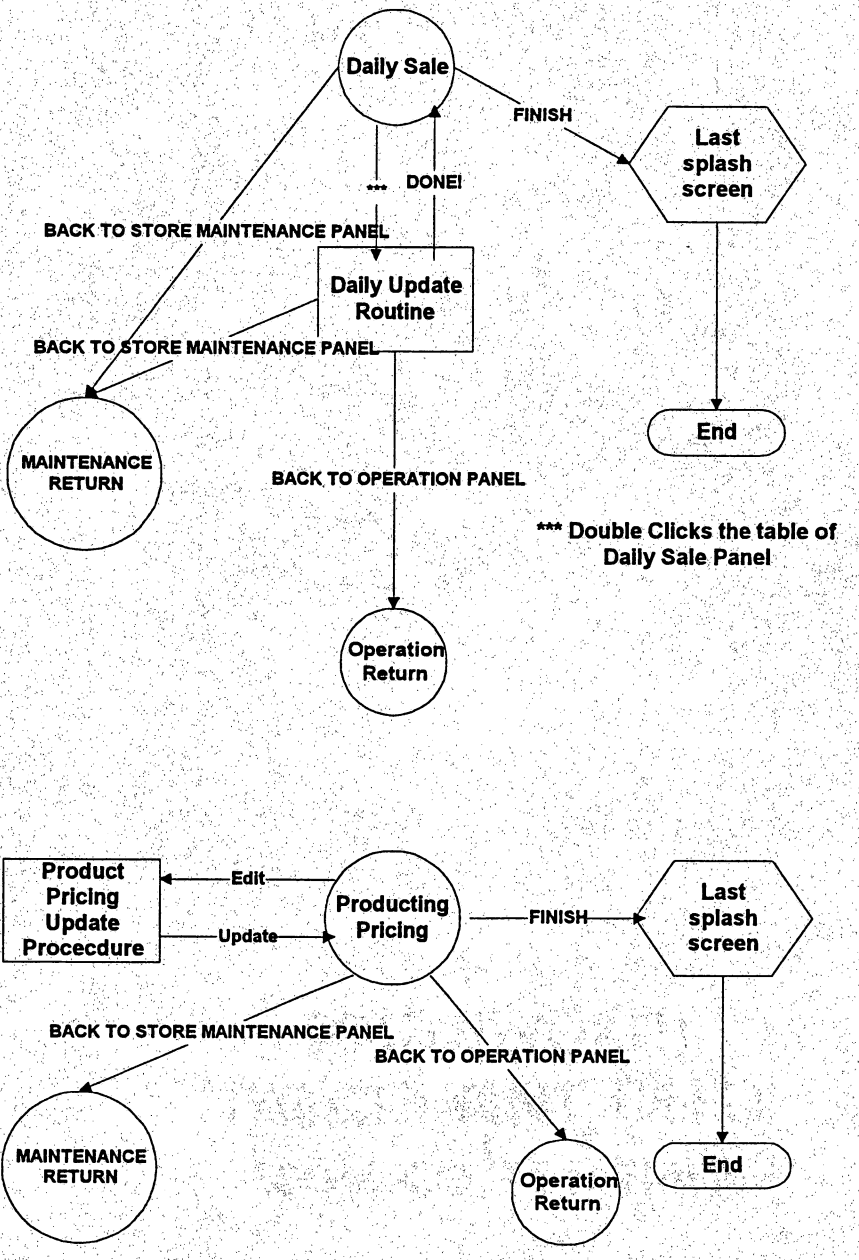


Figure 9.5 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

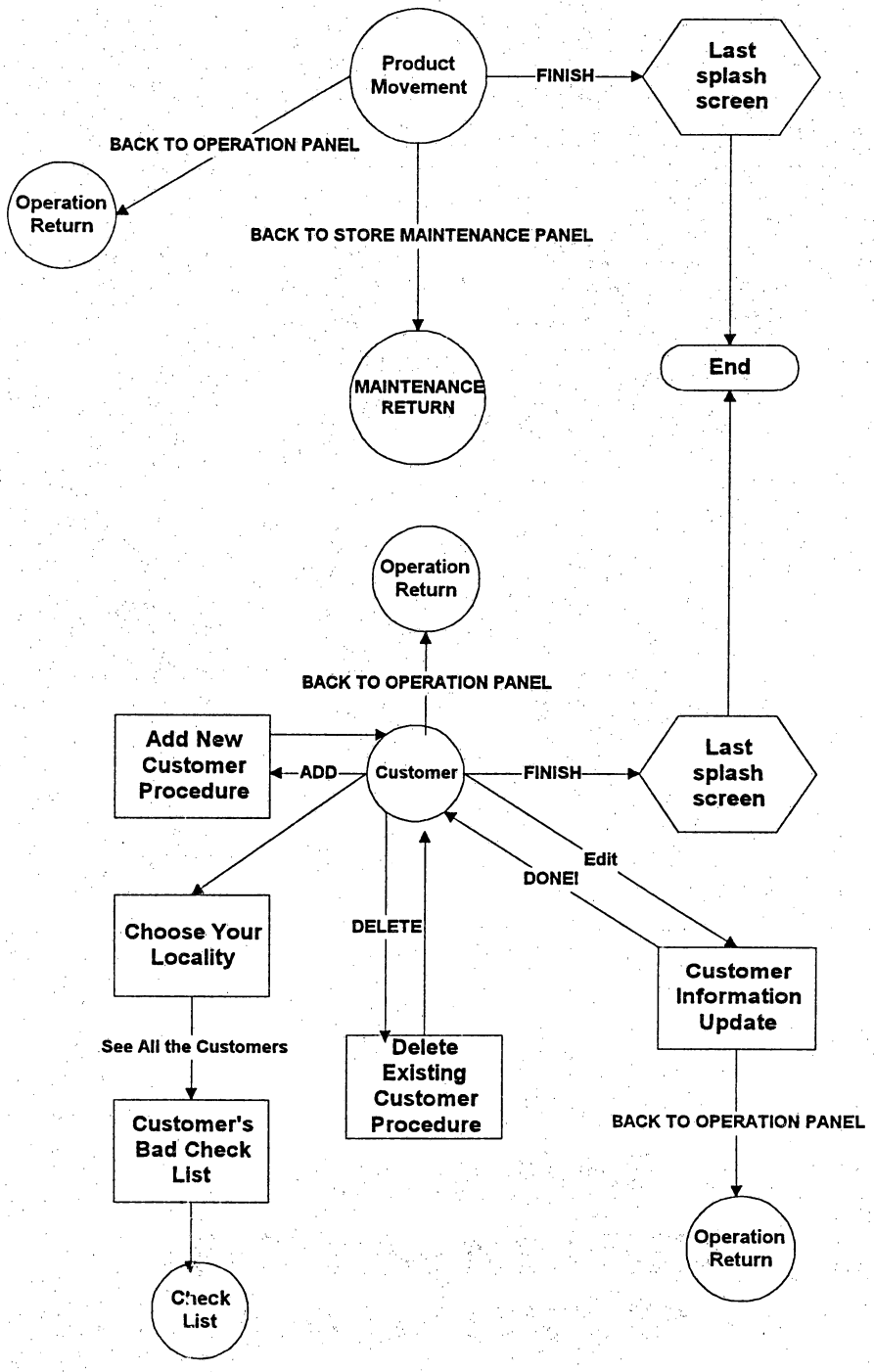


Figure 9.6 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

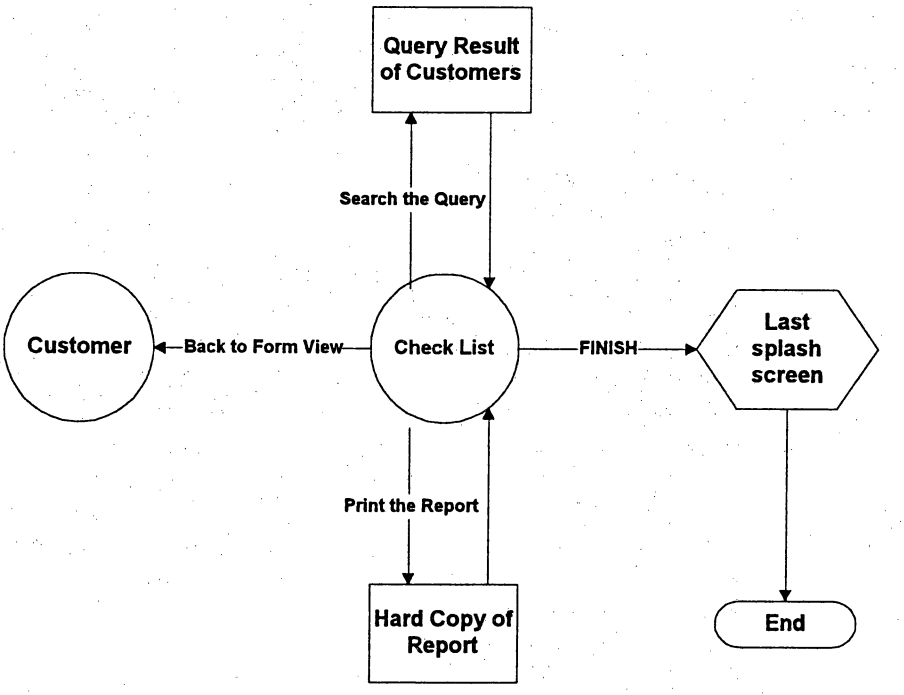
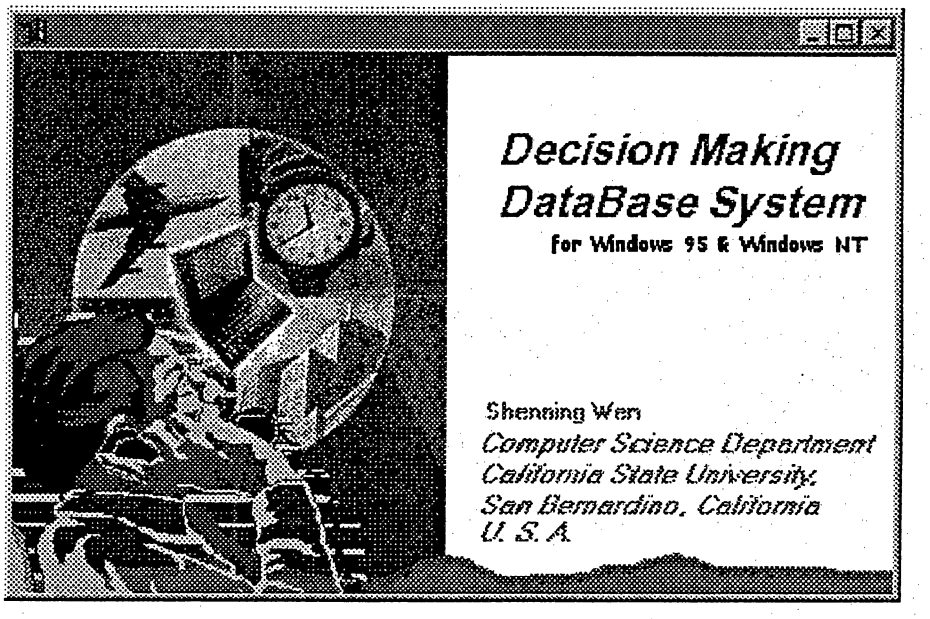


Figure 9.7 The Decision Tree of the Decision Making Database System(DMDS), *(Continue)*

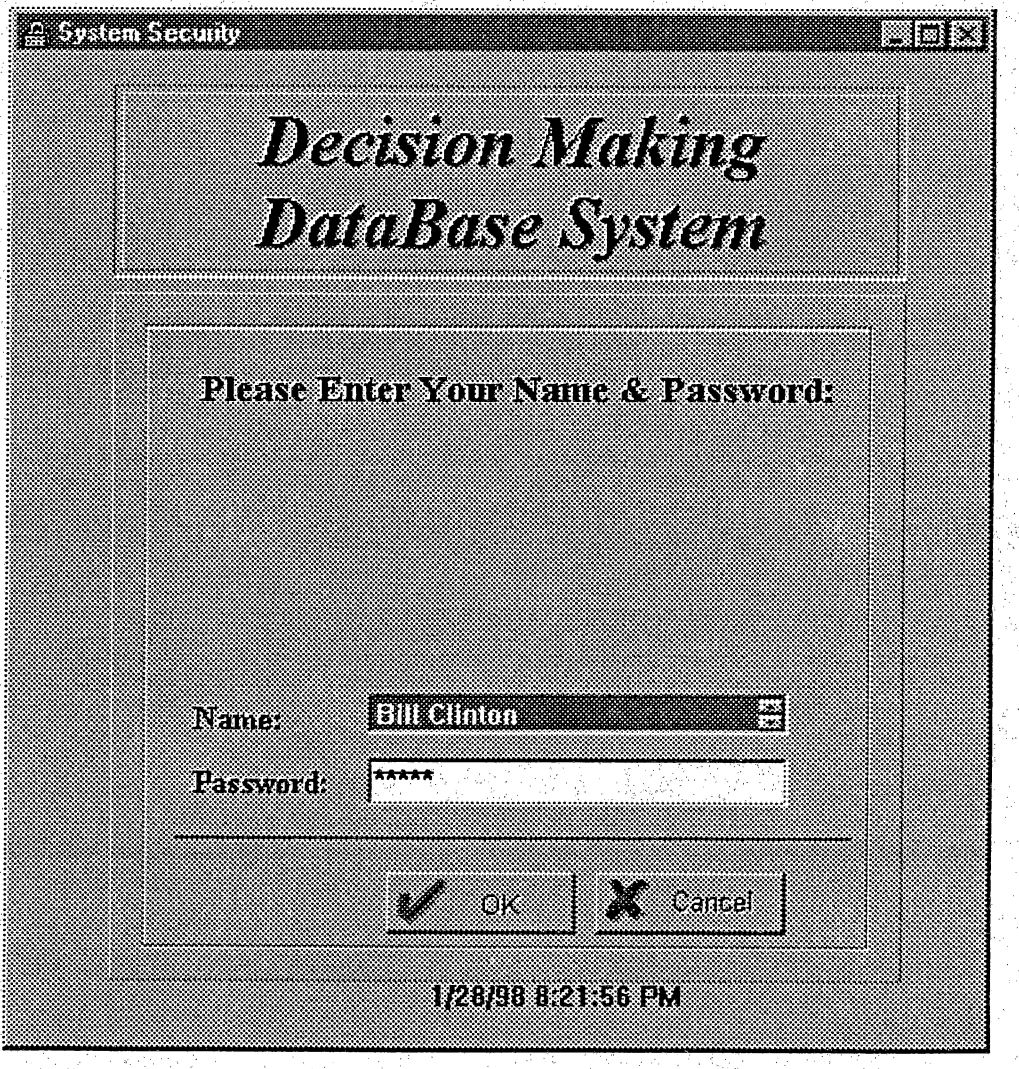
Access **Beginning Splash Screen (Automatically)**



Note

- This is the splash screen at the very beginning of DMDS software.

Access Beginning Splash Screen → System Security

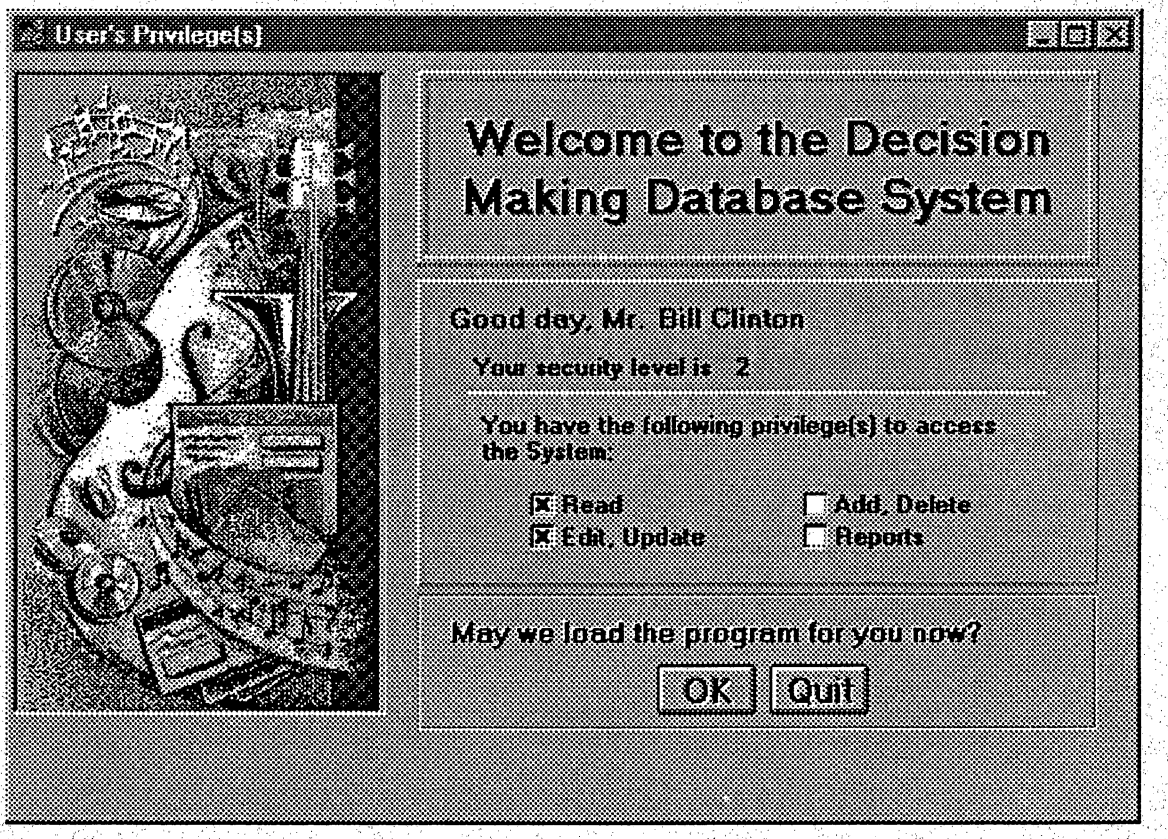


Notes

- This is the login screen for the DMDS system. The user can pick up his/her name from the **Name** drop down list box, and enter the corresponding authorized password for admittance. After entering the user's name and password, the command key **OK** should be pressed to submit the request.
- The **Cancel** key can be used to abort the login process.

- If the user fails to attempt to login the system for three times, the system will shut down.

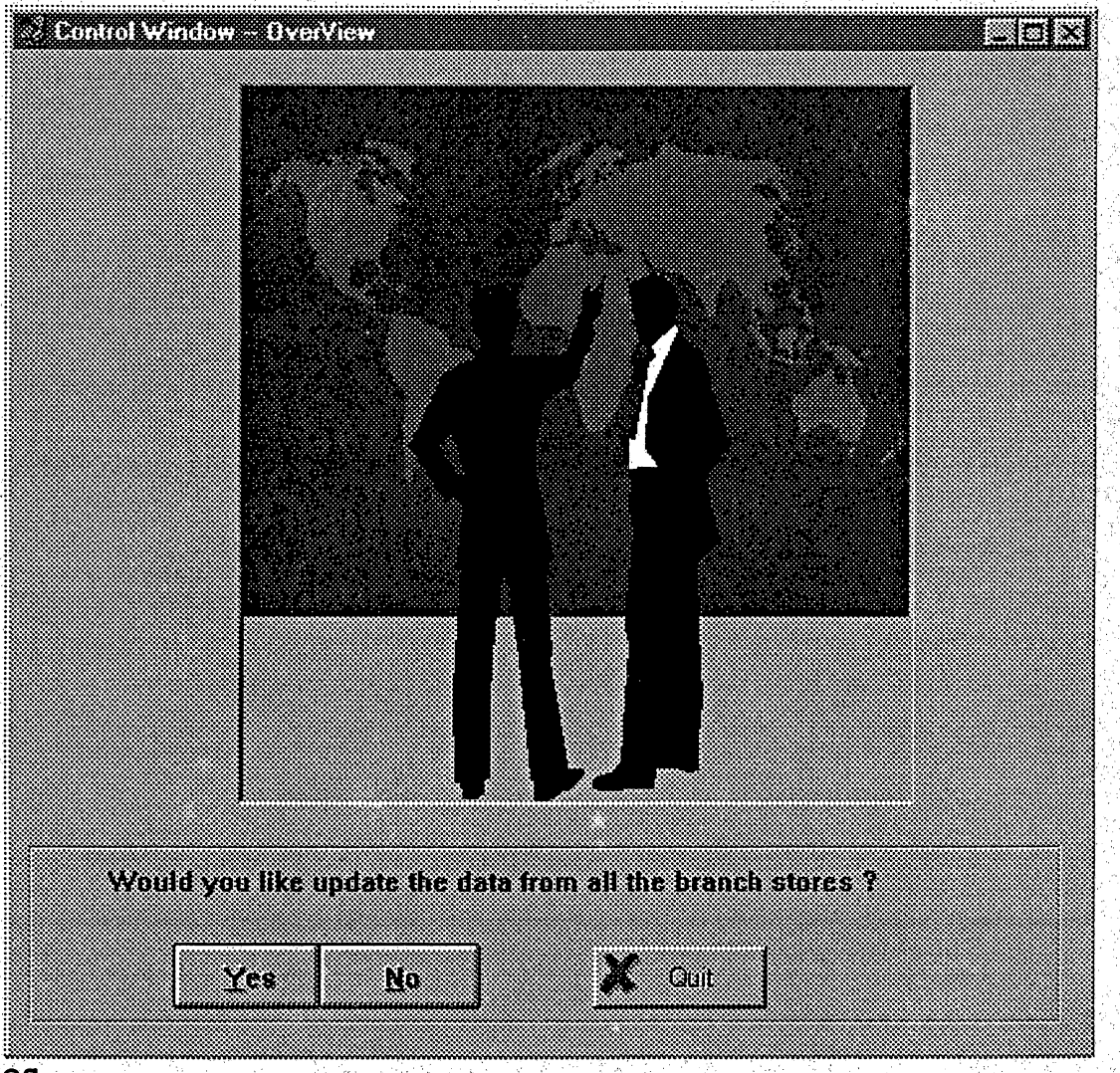
Access System Security → User's Privilege(s)



Notes

- There are four different security levels; each one has its own privilege(s) to access the system(please see page 55 of this report for details).
- In the lowest portion of this screen, the user has two options:
  - Press **OK** Key: The program and its associated data will load.(the progress indicator will show)
  - Press **Quit** Key : The program will lead the user to the last splash screen of the program.

Access User Privilege(s) → Control Window--OverView



Notes

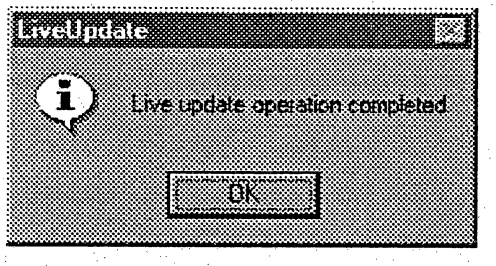
- On this screen, the user has three options to decide whether to run the live update from all the branch stores.
- The three options are:
  - **Yes** Key: The system will go ahead to update the information from all the branch stores(simulation will occur).
  - **No** Key: No system update is needed to this operation(i.e. skip this update procedure).
  - **Quit** Key: User chooses to terminate the program.

Access Control Windows-OverView ⟶ LiveUpdate

Notes

- On this screen, the simulation of live-update operations is in progress. The progress indicator and its corresponding status are shown to the user.
- **Next** Key: It will forward to next screen to view the update information after the update process is completed.

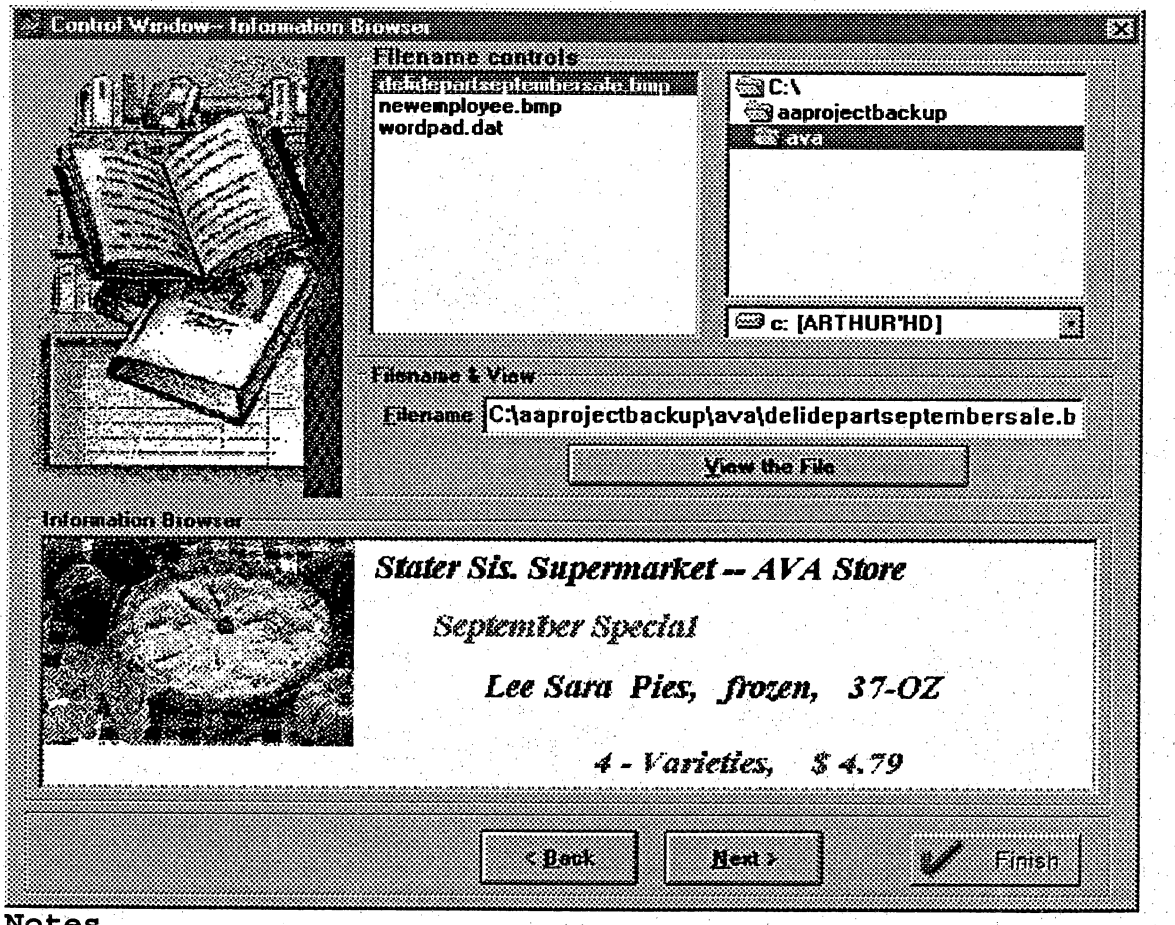
Access **LiveUpdate**



Notes

- This is the information message when all the live update operations are complete. The purpose of this message is to give the user peace of mind after doing other things during the prolonged update process.
- After the **OK** key is pressed, the three command options will be enabled.

Access LiveUpdate ⟶ Control Window-Information Browser

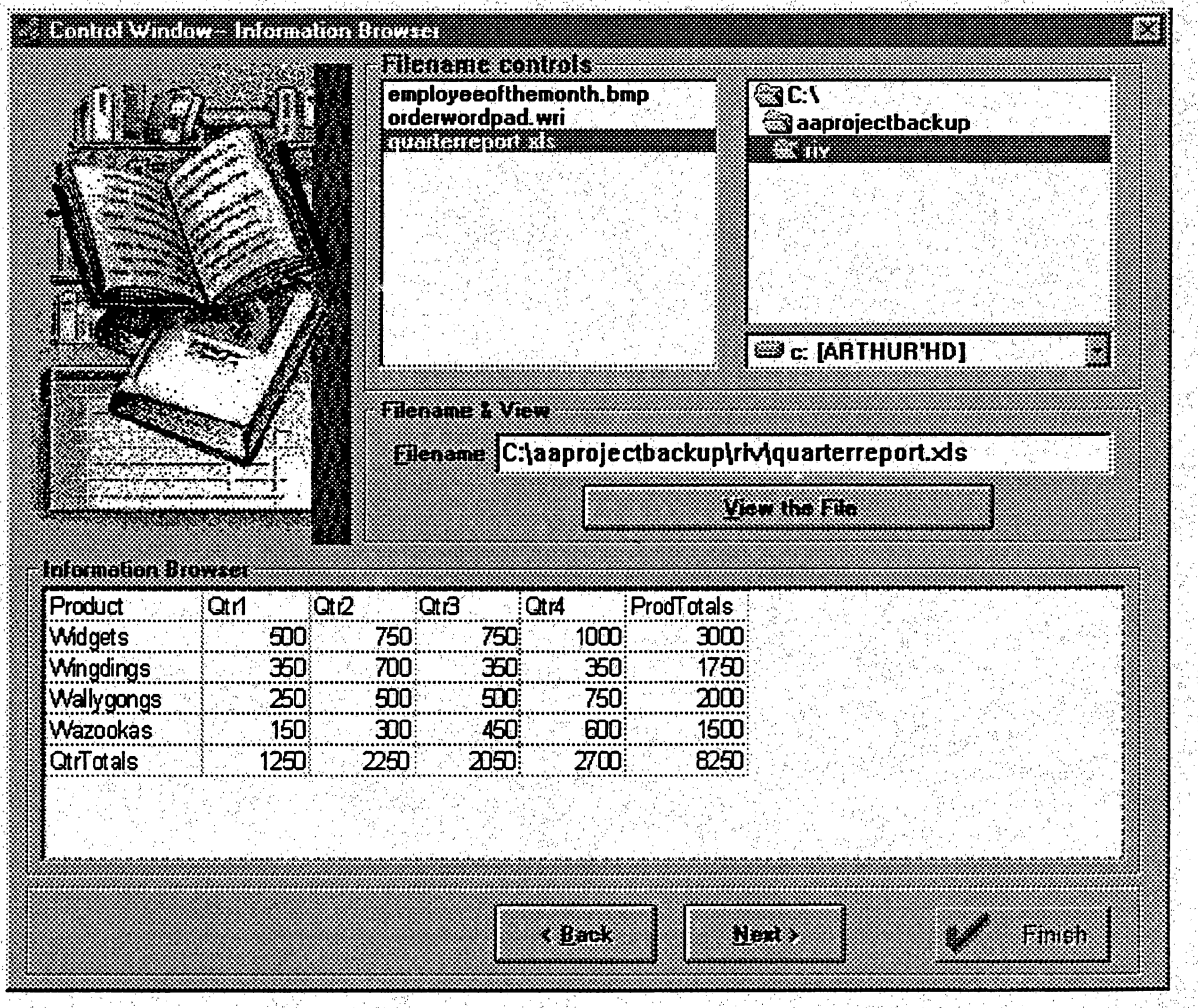


Notes

- This is the screen where the user can view the updated information from the *live update screen*.
- First, the user needs to highlight the default directory, which stores the update information(i.e., c:\aaprojectbackup, in this case), and to choose the target branch store to view.
- Second, from the **Filename controls** box, choose the filename the user wants to see. The chosen file will show the whole path in the **FileName** Window, then the user is asked to press the **View the File** command button to see the content of the file.
- Finally, the content of the file is shown through the **Information Browser** window.

- If the user wants to view the full size of the file or do some modification, the user double clicks the file itself, which leads the user to the corresponding application, which hosts it.
- There are three options:
    - **Back** Key: It will lead the user back to *LiveUpdate screen*.
    - **Next** Key: It will lead the user forward to *Select-an-Operation screen*.
    - **Finish** Key: It will terminate the program.

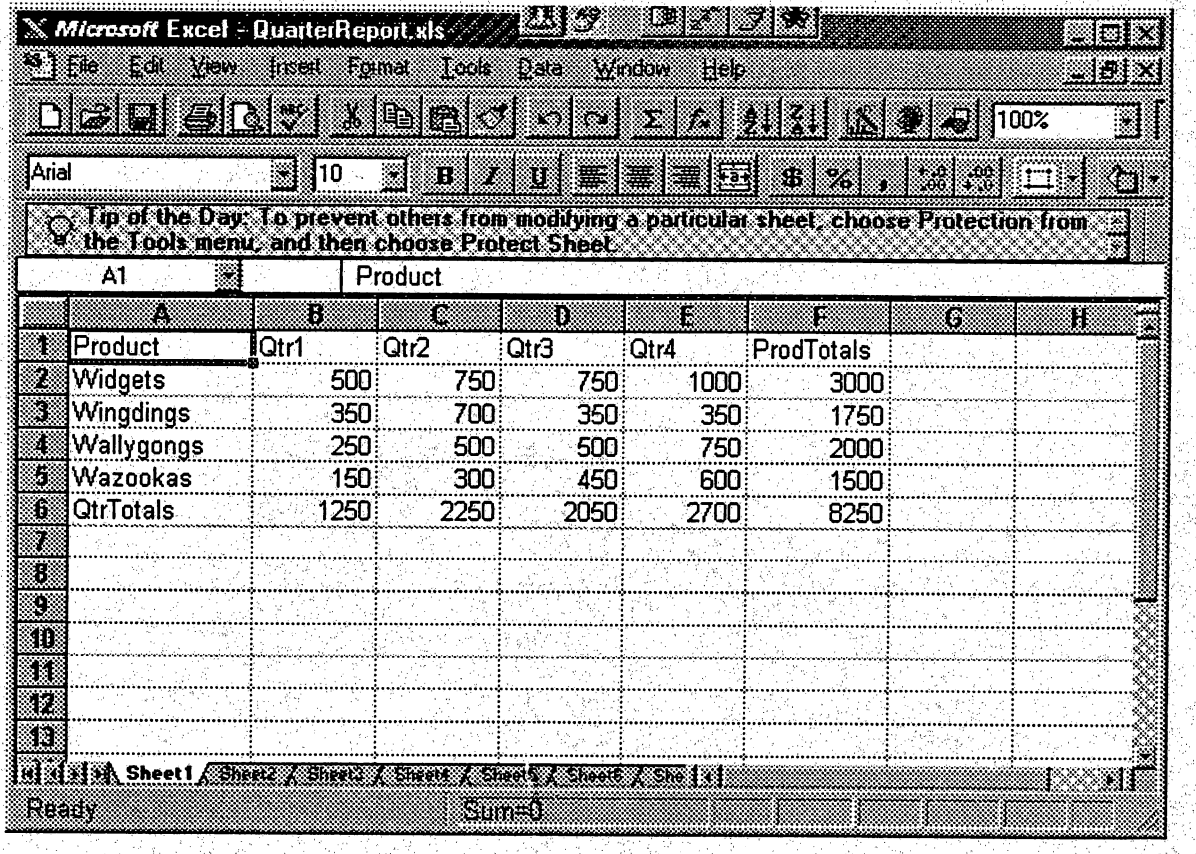
Access LiveUpdate Control Window-Information Browser



Notes

- On this screen, the Microsoft Excel spreadsheet is the host program for this quarter sale report. In the **Information Browser** window, the partial quarter sale report is shown.
- If the user want to modify the quarter sale report, a double-clicks the report itself will lead the user to the Excel spreadsheet Program (this can be seen from the next page).

Access LiveUpdate → Control Window-Information Browser

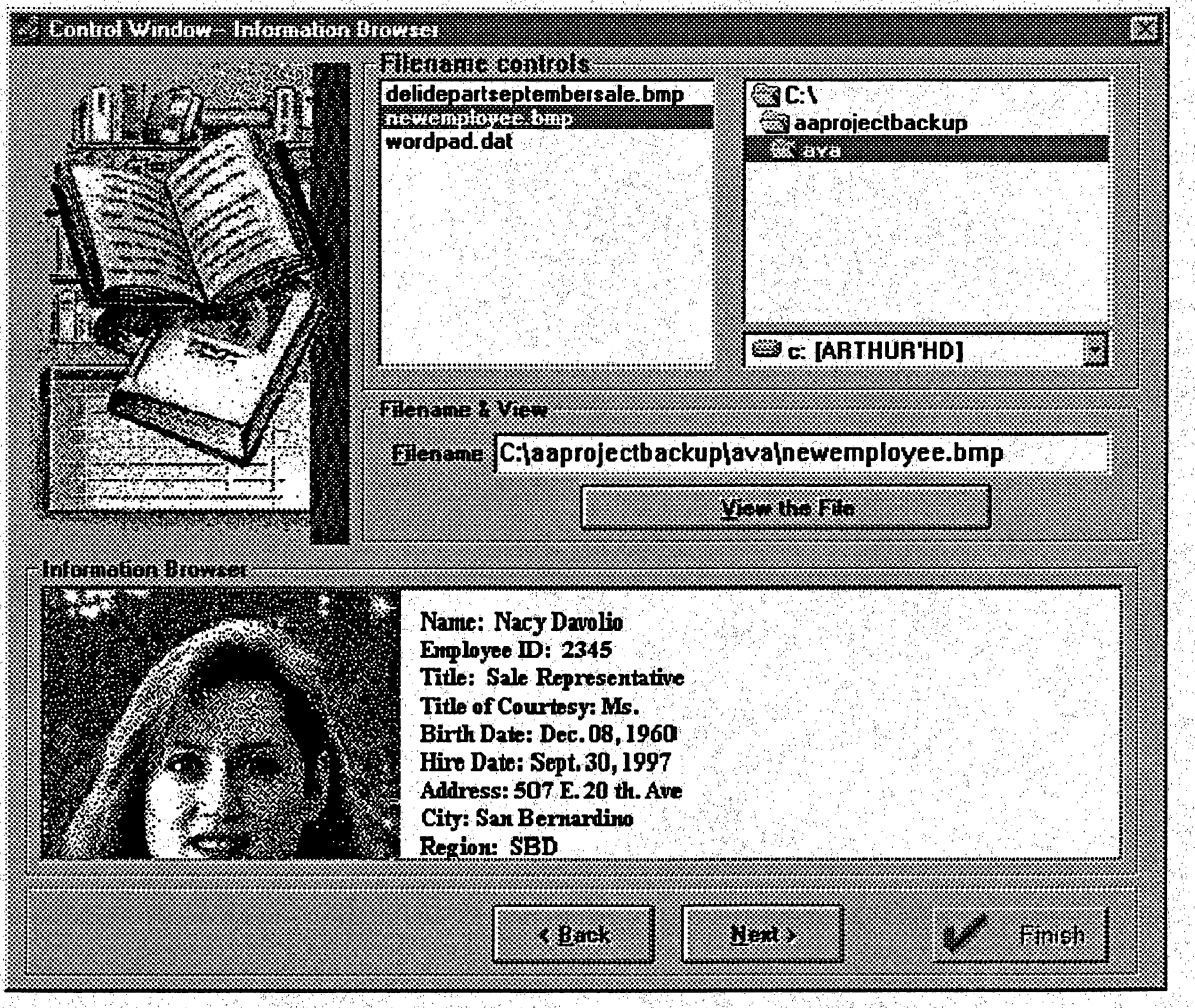


	A	B	C	D	E	F	G	H
1	Product	Qtr1	Qtr2	Qtr3	Qtr4	ProdTotals		
2	Widgets	500	750	750	1000	3000		
3	Wingdings	350	700	350	350	1750		
4	Wallygongs	250	500	500	750	2000		
5	Wazookas	150	300	450	600	1500		
6	QtrTotals	1250	2250	2050	2700	8250		
7								
8								
9								
10								
11								
12								
13								

Note

- This is the continuation of previous screen. A full features of Excel spreadsheet program is loaded for the user's modification.

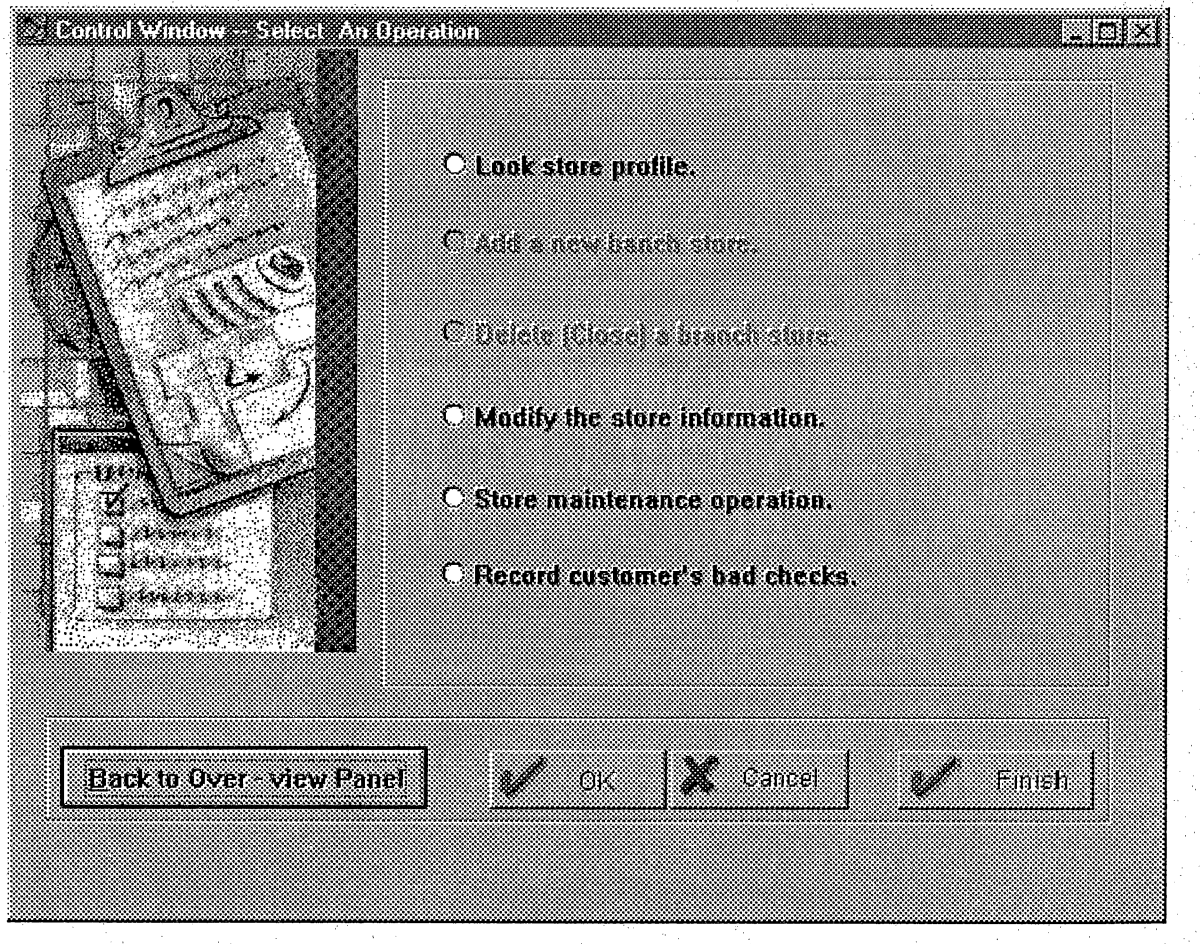
Access LiveUpdate → Control Window-Information Browser



Notes

- This is another snap shot of **Information Browser** showing the Employee profile of AVA store in *.bmp format.
- Using the OLE technology in this module, a wide variety of file formats can be accepted.

Access Information Browser ⟶ Select-An-Operation

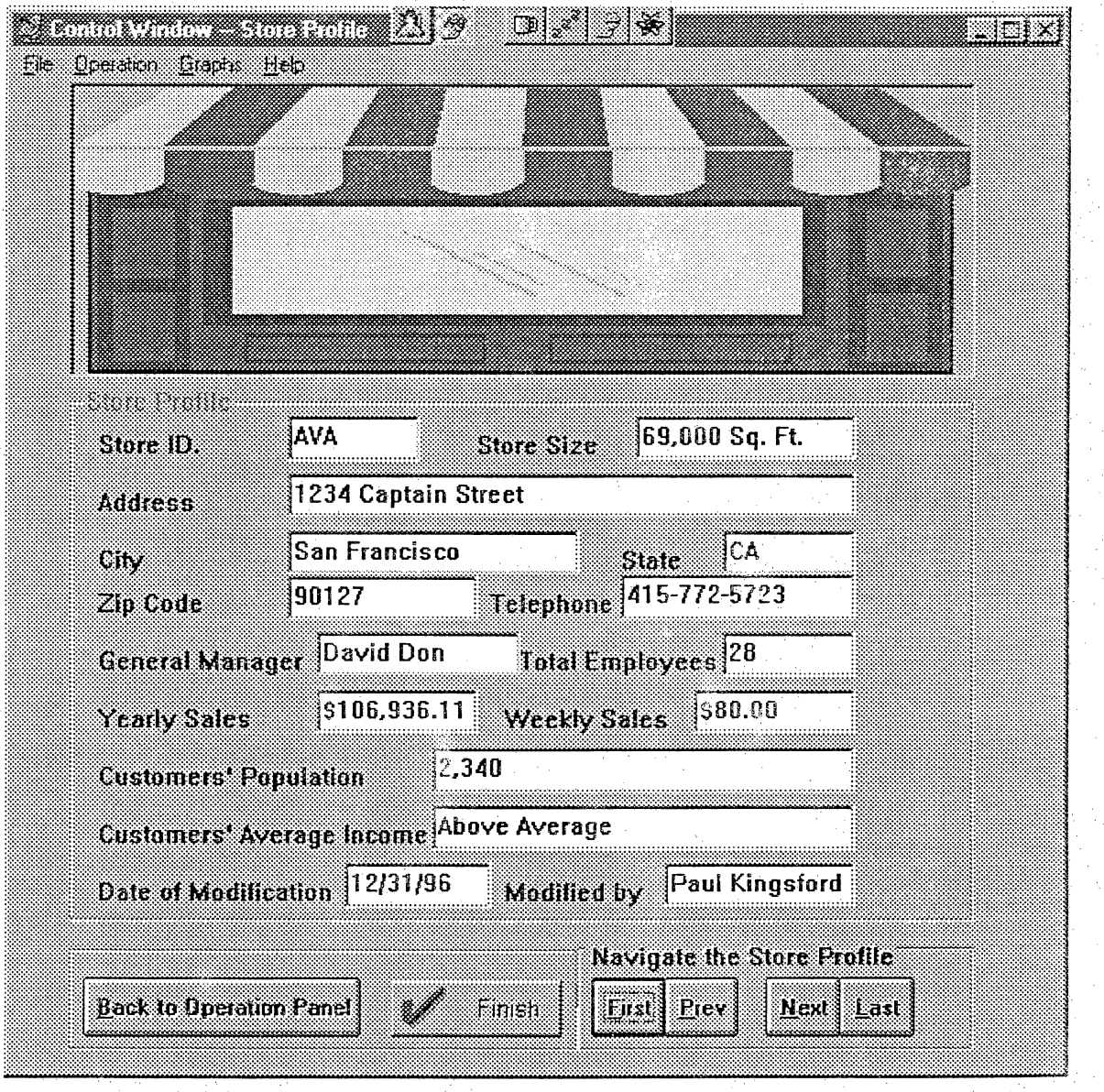


Notes

- On this screen, six operational options are listed for the user's command. Since the user(Bill Clinton, in this case) has security level 2, his privileges for the operations are limited to Edit, Update, and Read. So, two of the operational procedures are dimmed(disabled) in this panel to prevent such unauthorized usage.
- Only the users who have the highest security level(i.e., level 4) can have full access to all operations in this panel.
- In this screen, the user can only pick one operation to execute at a time. After the user chooses the operational item, the **OK** Key should be pressed to submit the request.
- The other three command buttons function as follows:

- **Back to Over-view Panel** Key: It will go back to the previous ***OverView*** control window.
- **Cancel** Key: It will cancel the action of **Select-an-Operation**.
- **Finish** Key: It will terminate the program.

Access Select-An-Operation ___ (Look Store Profile) Store Profile



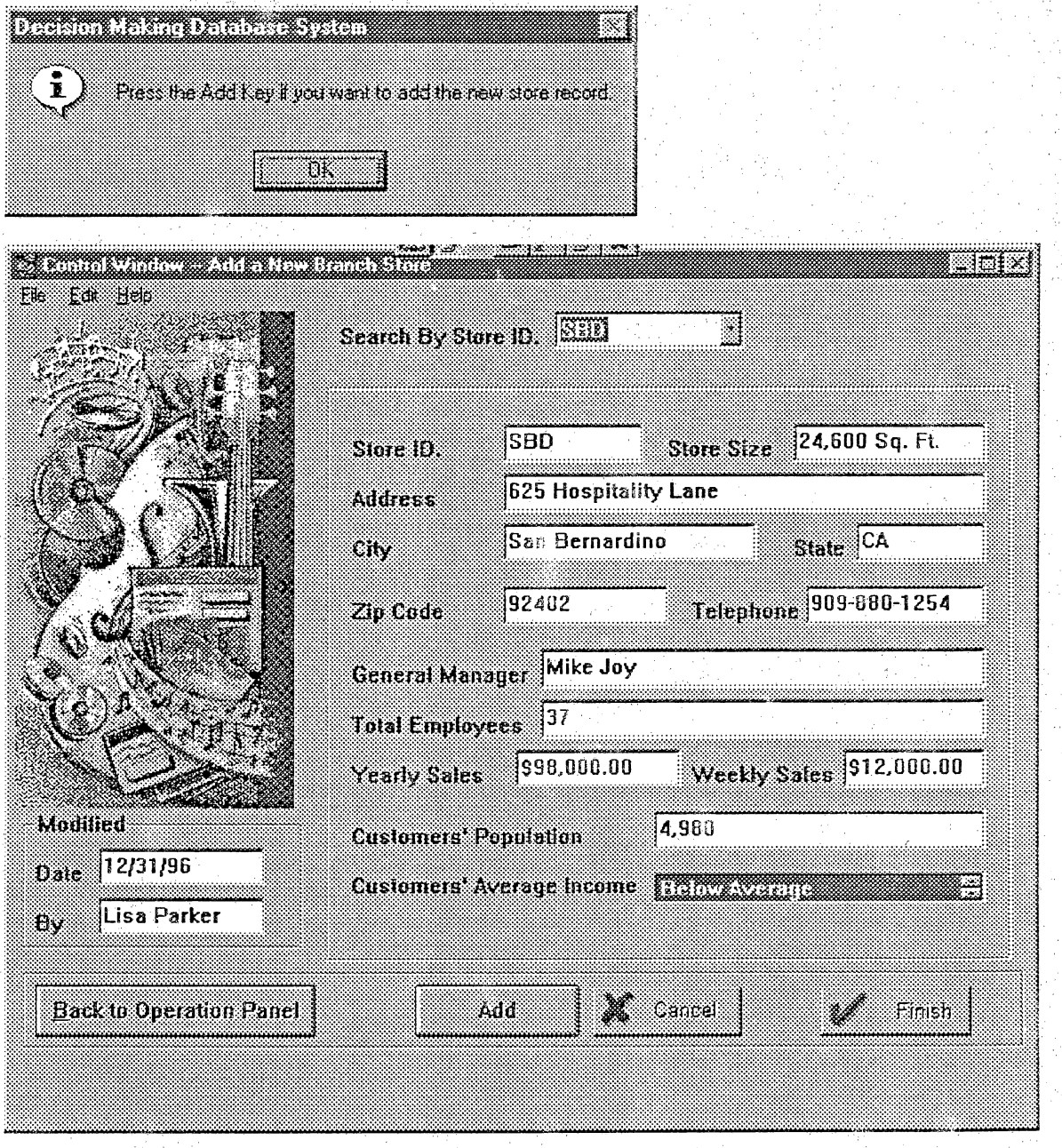
Notes

- On this screen, the AVA store profile is presented. The pull down menu and command buttons were added for the convenience of the user.
- In the ***graphs*** of the pull down menu, there are three sub-menu items: *Sales by Region*, *Sales by Month*, and

Sales by Store. In each sub-menu item, the graph further divides into two kinds of graphic representations: *pie chart*, and *bar chart*.

- For easy navigation of the store profile, four command buttons function as follows when pressed:
  - **First** Key: The first record of the store profile data set will present.
  - **Prev** Key: The previous record of the store profile will come into sight. If the current position is the first record of store profile, the information shown in the window will not change.
  - **Next** Key: The next record of the store profile will come into sight. If the current position is the last record of store profile, the information shown in the window will not change.
  - **Last** Key: The last record of the store profile data set will present
- In the bottom left corner there are two command keys which function when pressed:
  - **Back to Operation Panel** Key: It will lead the user back to *Select-An-Operation* panel.
  - **Finish** Key: It will terminate the program.

Access Select-An-Operation ⟶ (Add a new branch store)

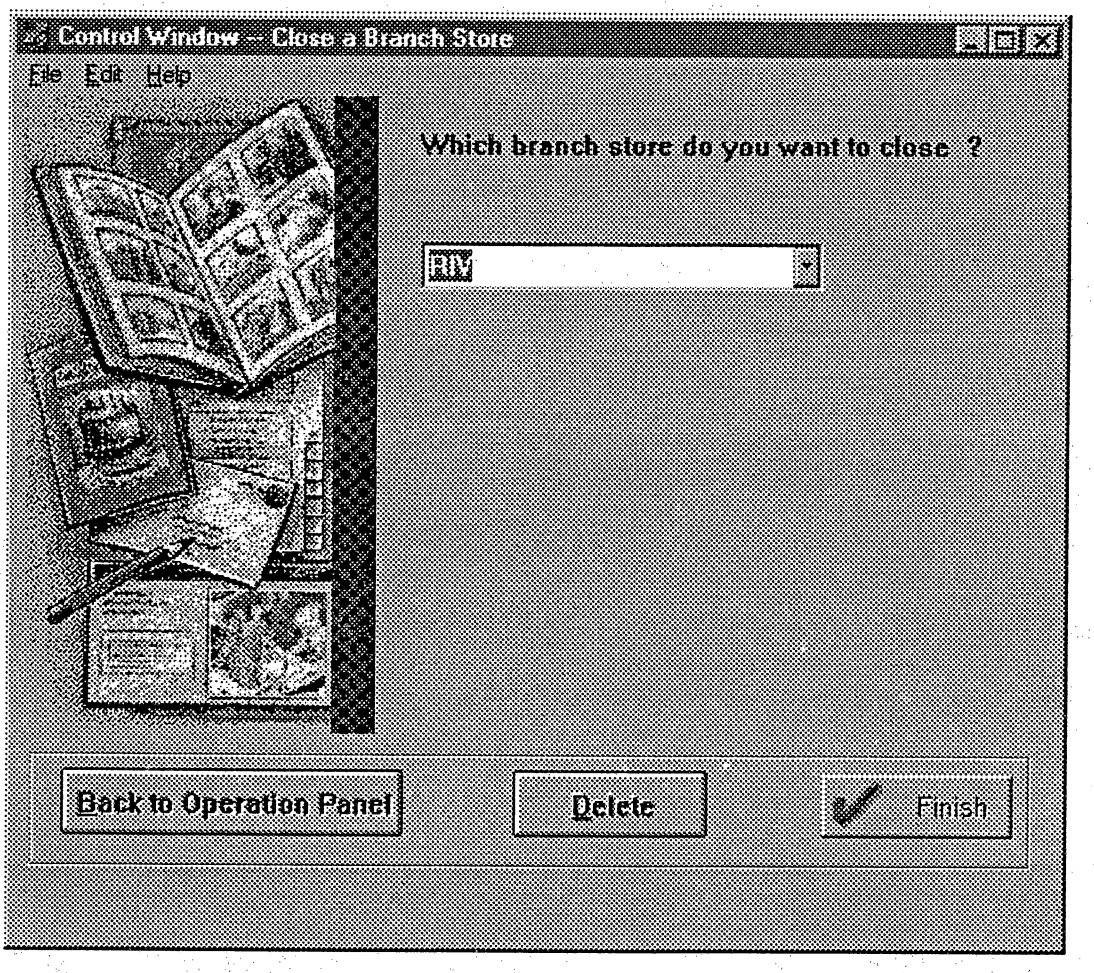


Notes

- It is a good idea to give the user some hints through the **information box** on how to add the new store information to the store profiles' database.

- There are four command buttons available on this screen:
  - **Back to Operation Panel** Key: This key will bring the program back to the ***Select-An-Operation*** panel for the next operation.
  - **Add** Key: This key must be pressed before the addition procedure begins.
  - **Cancel** Key: This key is needed in case the user decides in the middle of the process to abort the new store operation.
  - **Finish** Key: This key will lead the program to its final screen.

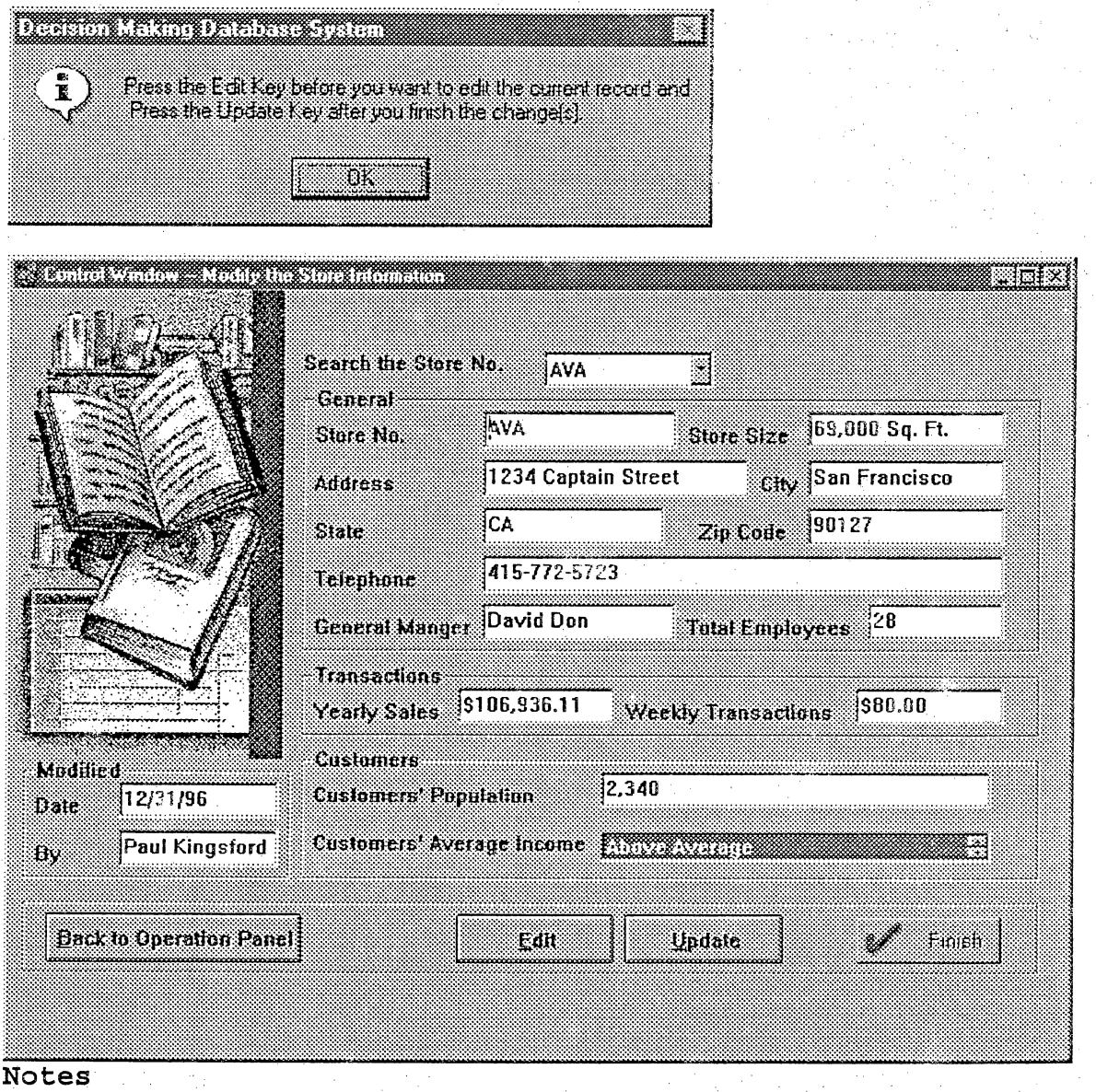
Access Select-An-Operation →(Delete(Close)a branch store)



Notes

- On this screen, the user can delete a store which no longer exist.
- First, from the drop-down box the user picks the name of the branch store to be deleted.
- Next, the **Delete** Key should be pressed. The system will then give the user a confirmation message box(not shown in this page) to warn the user about the result of such an operation. If the user agrees the operation is OK, the store information will be permanently removed from the database.

Access Select-An-Operation → (Modify the store information)

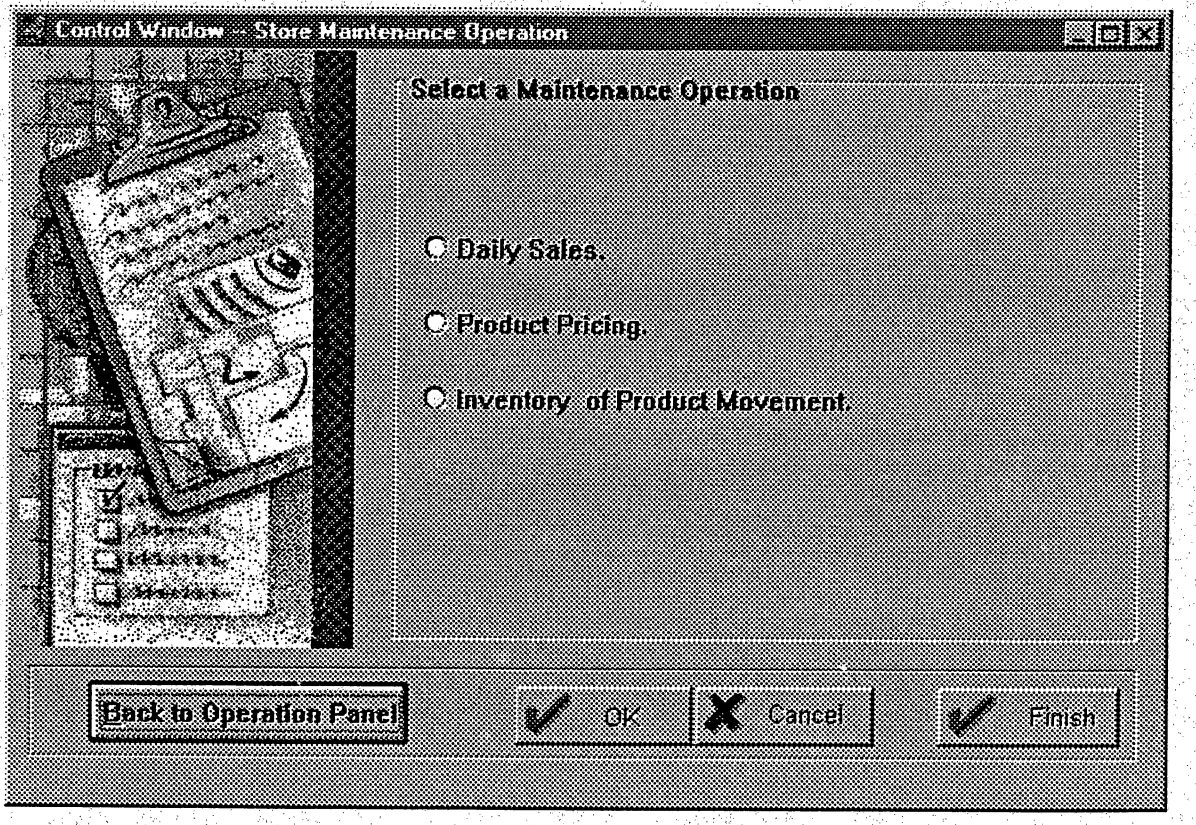


Notes

- It is necessary to give the user instruction through the **information box** on how to modify the store information before the store profile becomes editable.
- There are four command buttons available on this screen:
  - **Edit** Key: This key must be pressed before the edit procedure begins.

- **Update** Key: This key is needed to validate updated information(i.e., permanent saved) in the store database.
- **Finish** Key: This will lead the program to its final screen.
- **Back to Operation Panel** Key: This key will bring the program back to the ***Select-an-Operation*** panel for the next operation.

Access Select-An-Operation → (Store Maintenance Operation)

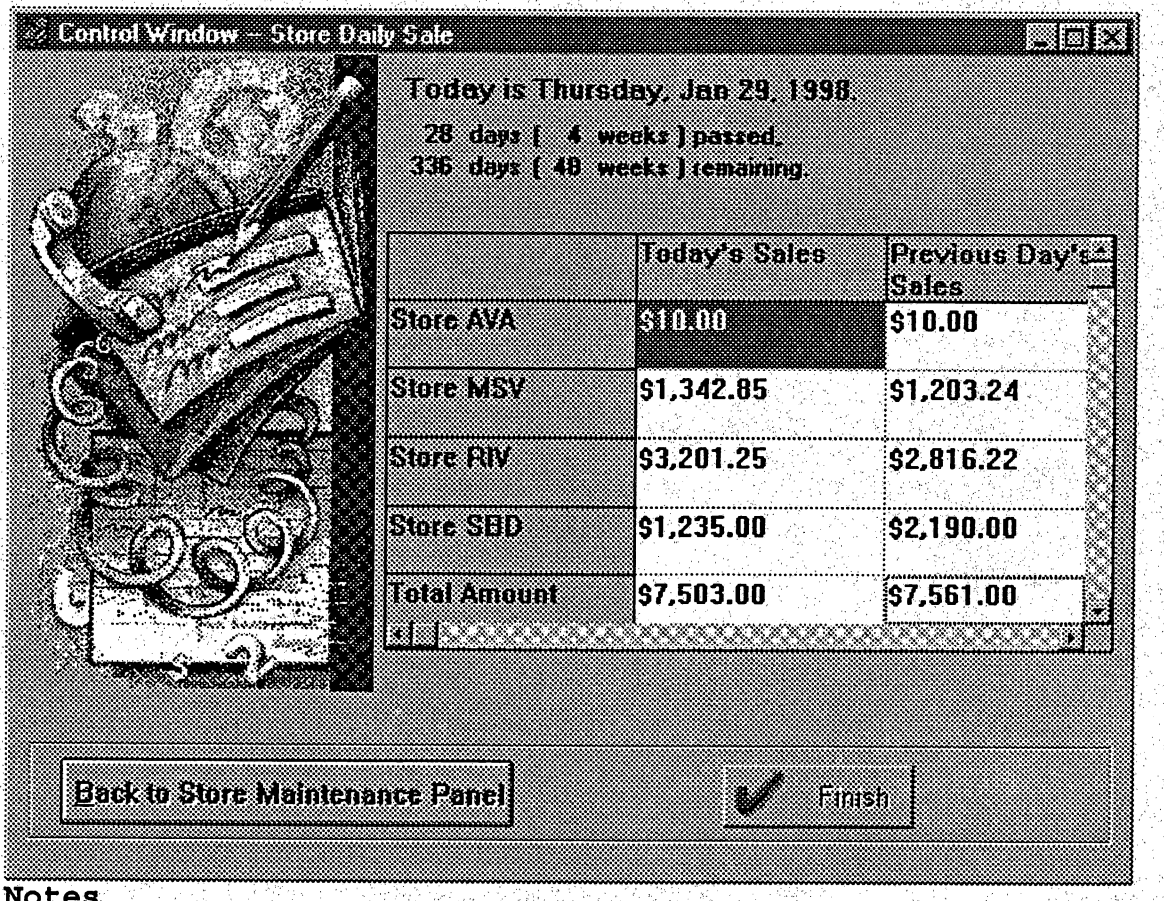


Notes

- On this screen the three most used store maintenance operation options are available for the user to choose from:
  - ***Daily Sales*** option: The daily sales'(previous day & today) transaction amount of all stores can be viewed through this option.
  - ***Product Pricing*** option: Three kinds of information(i.e., ***Product information***, ***Product pricing***, and ***Inventory Reports***) regarding each product will be available through this option.
  - ***Inventory of Product Movement*** option: The record of inventory products tracks the status of product stock in all the branches and assigns quota to all the stores of the new product.
- There are four command keys available on this screen , their functions are:

- **OK** Key: It is required for the user to press this key after each option is picked.
- **Cancel** Key: This key mainly serves to abort the action of the option chosen.
- **Finish** Key: This key will terminate the program.
- **Back to Operation Panel** Key: This key will bring the user to ***Select-An-Operation*** panel.

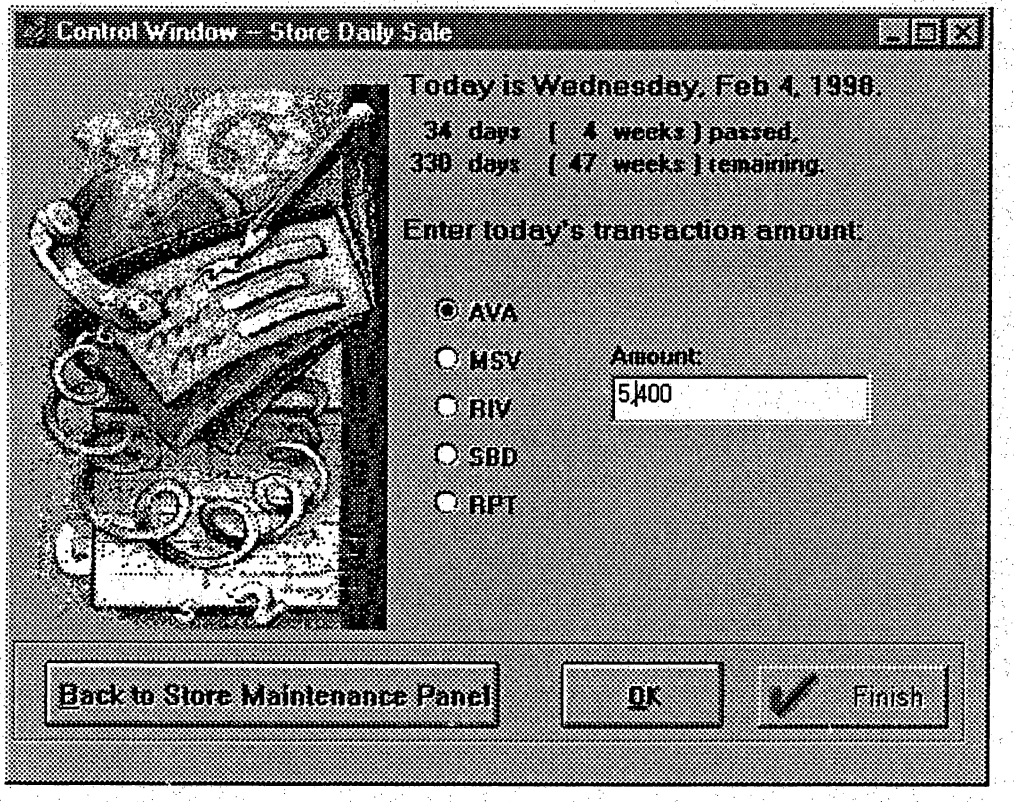
Access Store Maintenance Operation → (Daily Sales)



Notes

- On this screen, the daily transaction amounts of all stores are shown. If the user needs to record today's transaction amount for his/her store, he/she double click *today's sales* section for that store, and the ***Daily Update Routine*** screen will come into sight.
- There are two command keys available for the user:
  - **Finish** Key: It leads the user to terminate the program.
  - **Back to Store Maintenance Panel** Key: This key brings the user back to **the Store Maintenance Panel.**

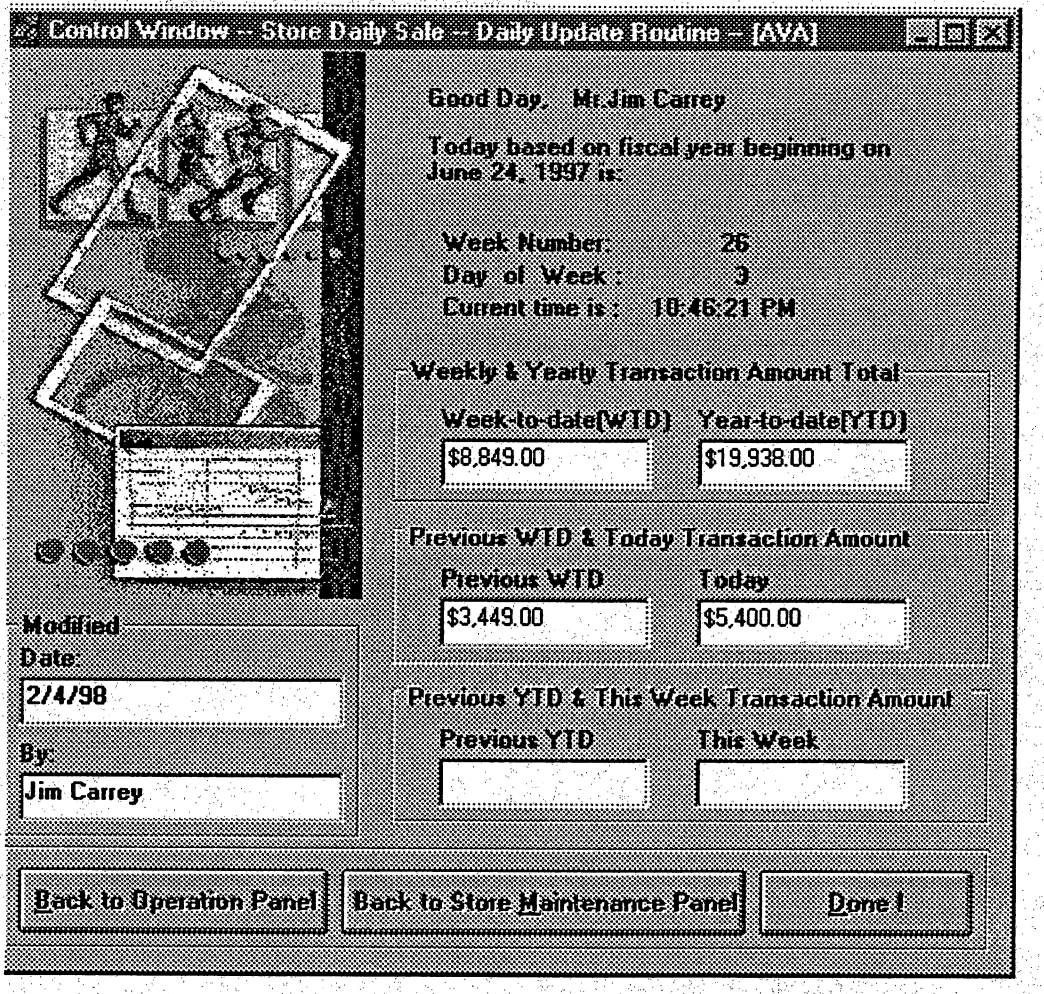
Access **(Double clicks the table of Daily Sales Panel)**



Notes

- On this screen, the daily transaction amount of the branch store can be written: First, the user needs to highlight the branch then enter the right amount of money for that store. The **OK** key needs to be pressed in order to validate this operation.
- The other command keys function as follows:
  - **Back to Store Maintenance Panel** Key: This key will bring the program flow to the ***Store Maintenance Panel*** for the next operation.
  - **Finish** Key: This key will terminate the program.

Access Store Daily Sale Daily Update Routine



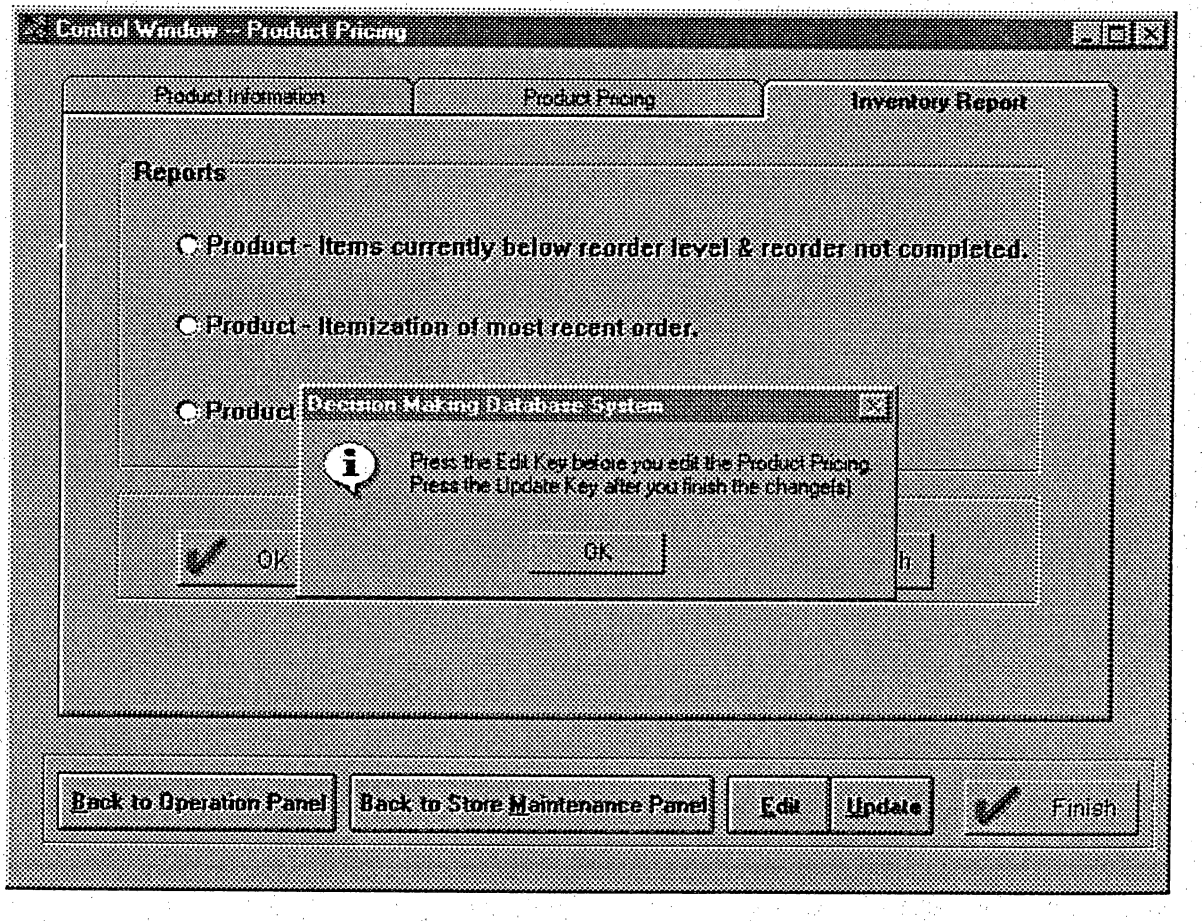
Notes

- On this screen, all the information related to daily, weekly, and yearly transaction amounts is shown through windows.
- Three command buttons are available for the operation:
  - **Back to Operation Panel** Key and **Back to Store Maintenance Panel** Key: These two keys lead the program flow back to the ***Select-An-Operation panel*** and the ***Store Maintenance Panel***, respectively.

- **Done !** Key: This key will let the user back to the ***Daily Sale Panel*** in case more data must be input to the system database.

→

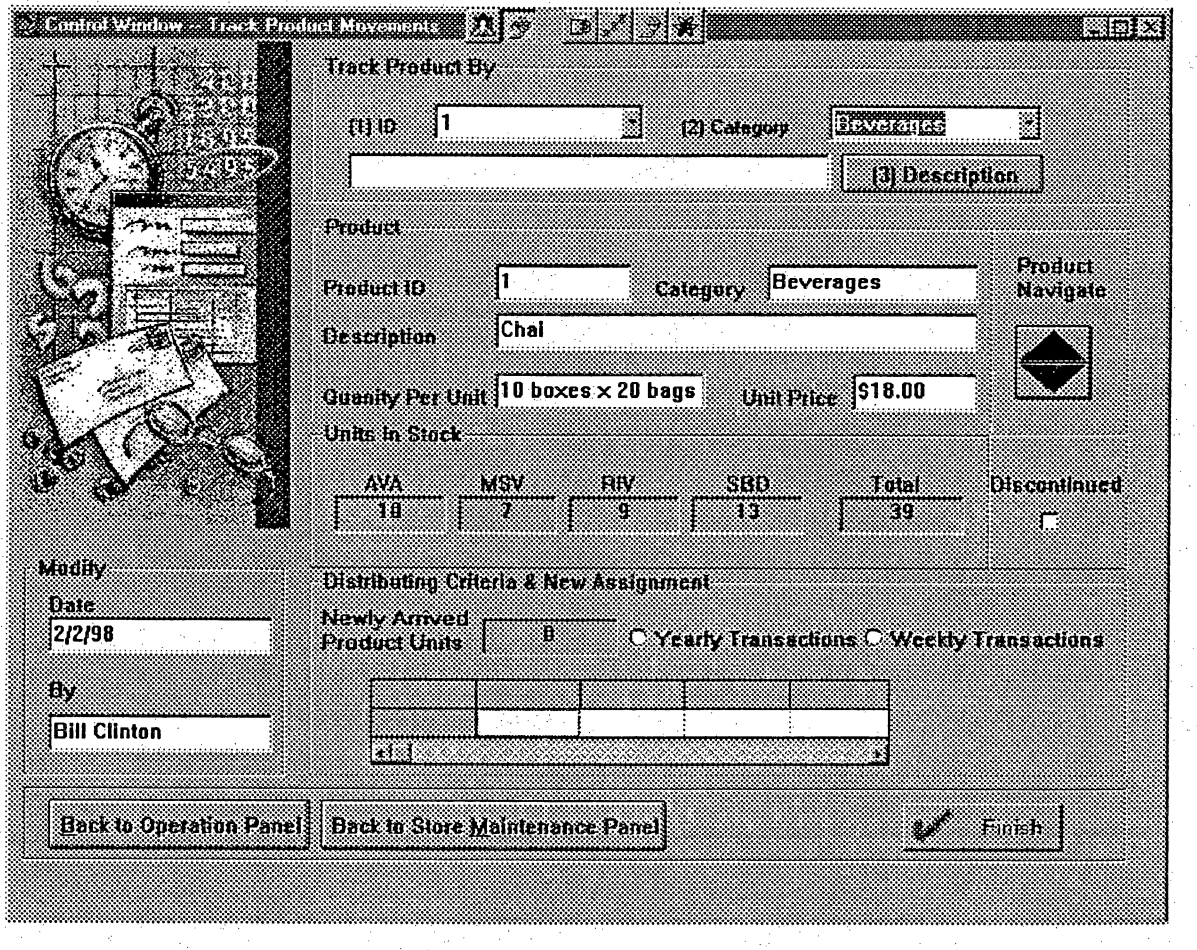
Access Store Maintenance Operation (Product Pricing)



Notes

- It is necessary to give the user instructions through the **information box** on how to modify the product pricing before product profile becomes valid.
- There are three options available in this index-card-like screen (i.e., ***Production information***, ***Product pricing*** and ***Inventory Reports***).
- Among the five command keys, the **Edit** and **Update** keys are enabled only in the ***Product pricing*** section. The rest of the three command keys(i.e., **Back to Operation Panel**, **Back to Store Maintenance Panel**, and **Finish**) are valid throughout all three sections.

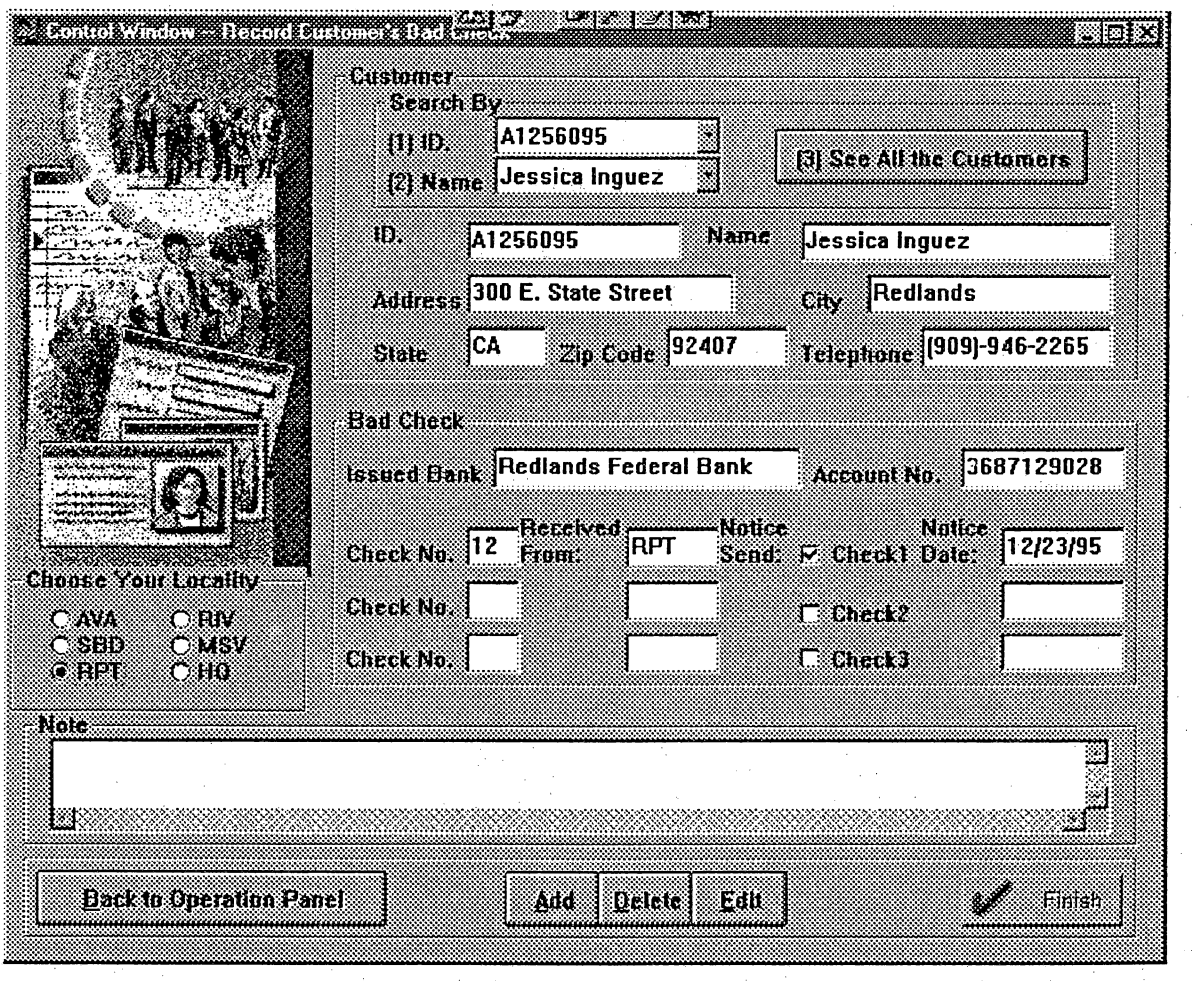
Access Store Maintenance Operation (Inventory of Product Movement)



Notes

- On this screen, first, the target product can be searched by **Product ID Number**, **Product Category** or **Product Title**. Also, this screen can let the user navigate the whole line of products through **Navigation up or down button**.
- Two of many features the user can benefit from are: the target product inventory status and distributing criteria for the newly arrived products (criteria can be changes according to weekly and yearly transactions).
- The functions of the three command keys of this screen have already been explained in the previous screens of this chapter.

Access Select-An-Operation → (Record customer's bad checks)

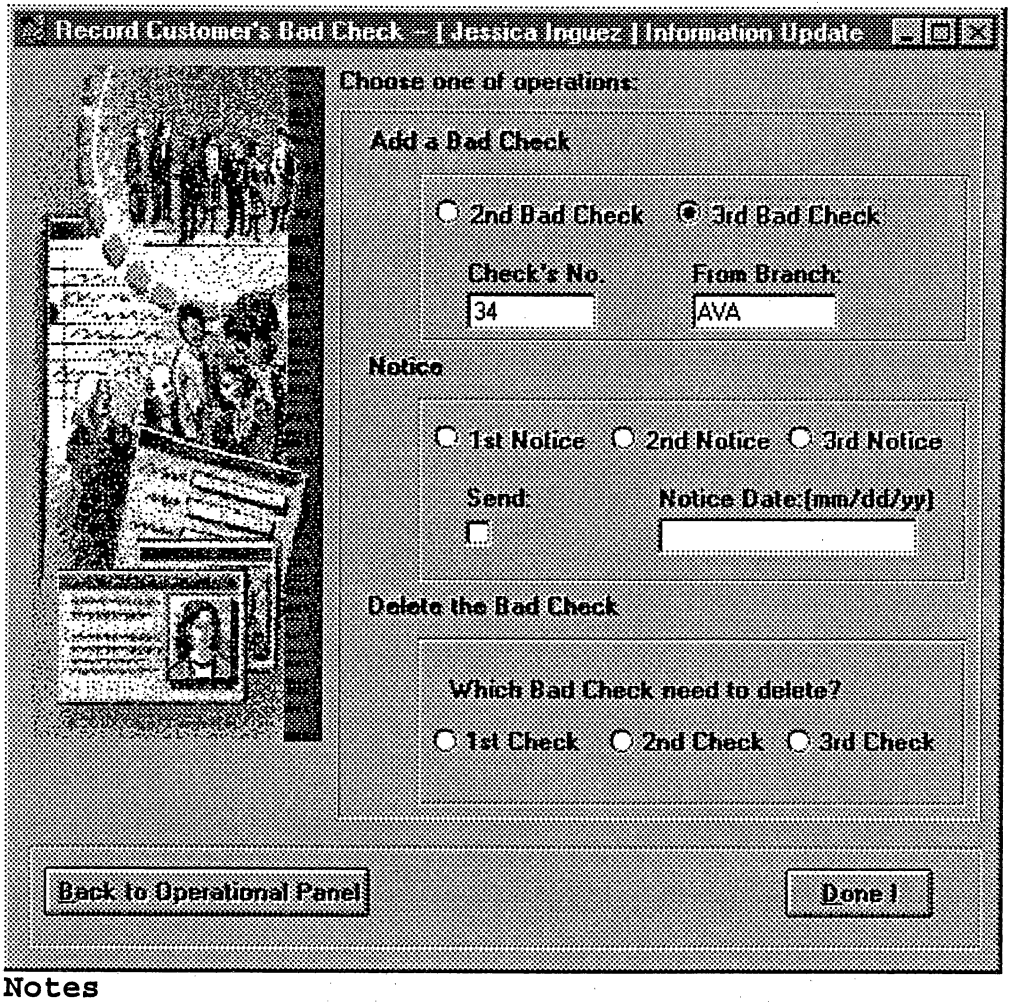


Notes

- On this screen, the bad customers' information can be viewed in two ways: one is from the local branch store's aspect (i.e., through the **Choose Your Locality** and the key -**See all the Customers**) and from the whole company customer database aspect (i.e., this can be done by choosing the **HQ** from **Choose Your Locality** and the key -**See all the Customers**).
- The target bad customer can be searched through Customer ID or by the Name of the customer.
- Three new command keys that need explaining are:

- **Add** Key: Using this key, the user can add the new bad customer record to the database.
- **Delete** Key: This key can delete the bad customer record that no longer has bad credit in the company.
- **Edit** Key: Edits the existing bad customer record.

Access **Record customer's bad checks** → **Edit Key**

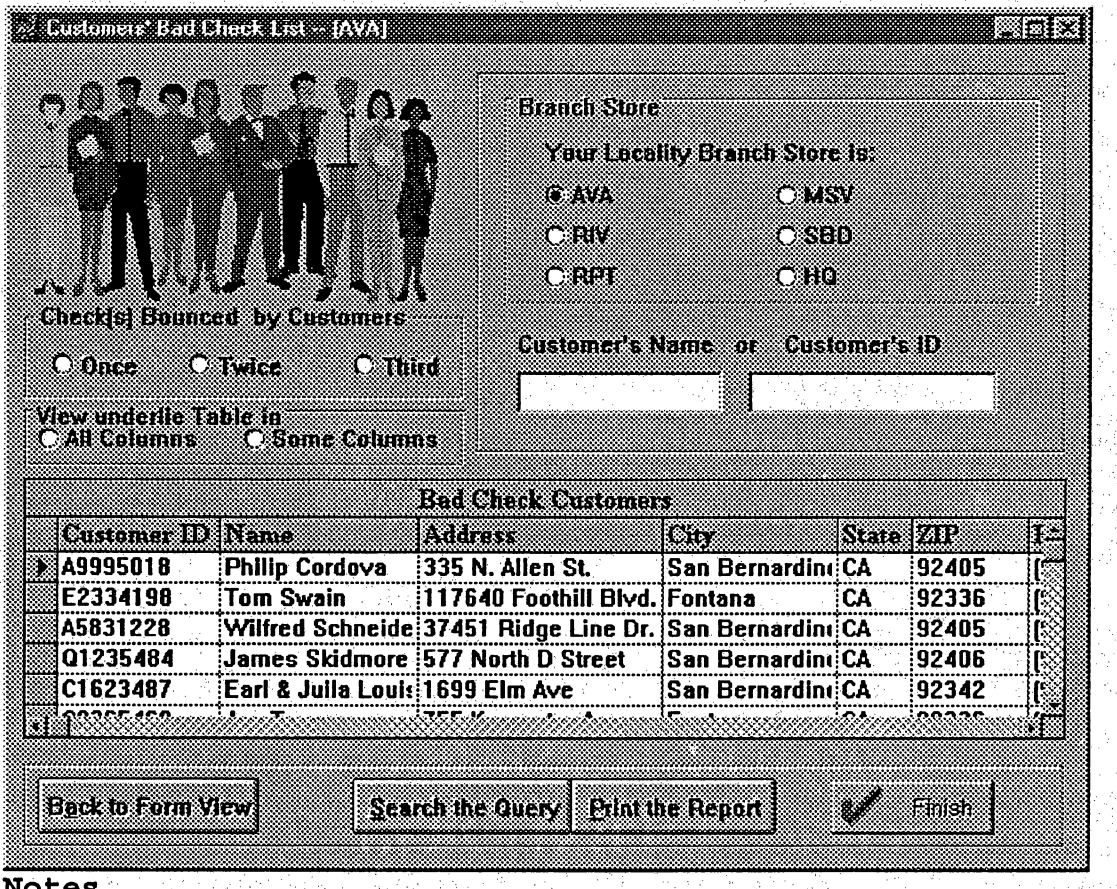


Notes

- On this screen, the customer's bad check information can be updated. This can be done in three ways:
  - **Add a bad check** section: First, enter the bad ***check's number*** and ***the branch store***, which received it. Second, fill in the number of bounced checks(i.e. second or third times).
  - **Notice sent information** section: First, click the *Send* box and write down the ***date*** of this notice will be sent; finally, record the number of notices sent (i.e. first, second, or third notice).
  - **Delete the bad check** section: This simply deletes the bad checks, which no longer exist in this particular customer's file.

- Two command buttons are available for the operation:
  - **Back to Operation Panel** Key: Back to the ***Select-An-Operation panel*** for the next operation.
  - **Done** Key: After the proper information is entered, this key serves to validate the updated information, and bring the program back to the ***Record customer's bad checks panel***.

Access Record customer's bad checks → Customers' bad check list

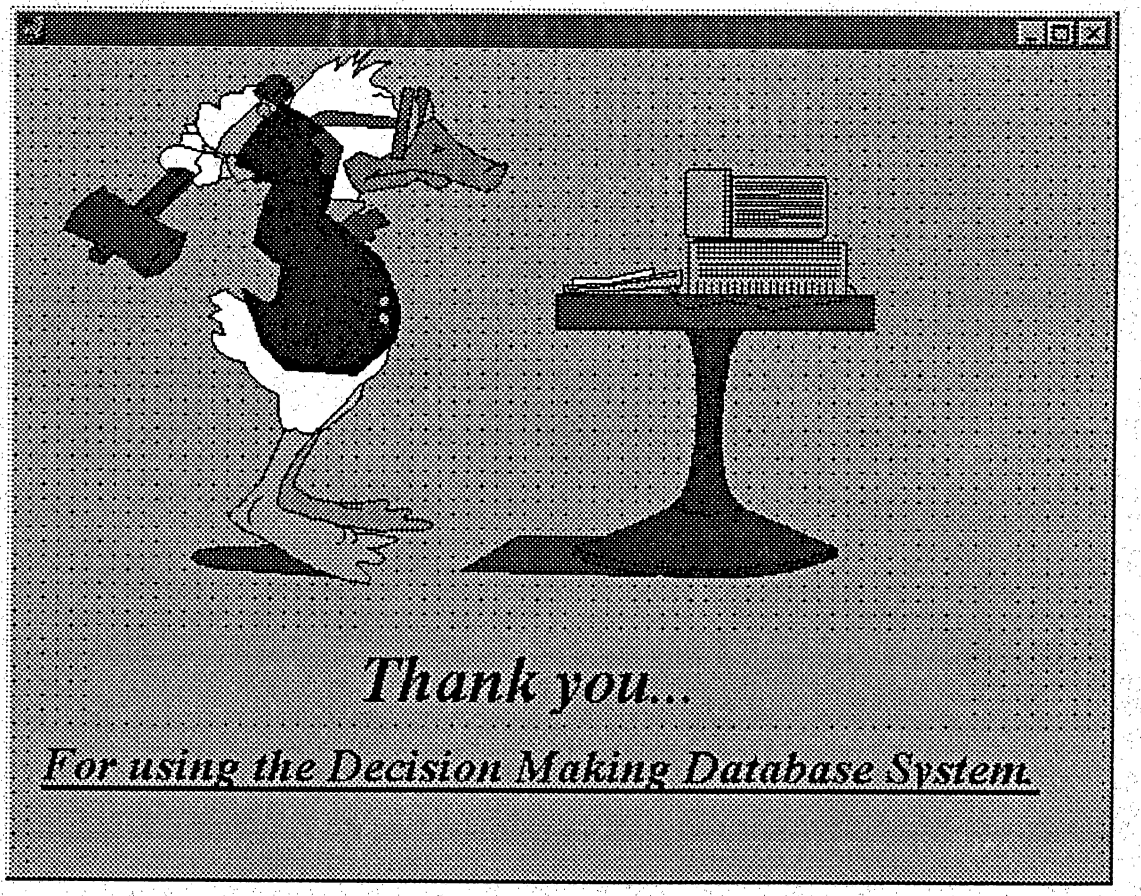


Notes

- On this screen, several options can be chosen to view the bad customers' records:
  - The customers can be displayed either from the local store's or from headquarters' point of view.
  - The customers can be searched by the **Customer ID** or **Customer's Name**.
  - The customers can be classified into three categories : one check bounced, two checks bounced, three checks bounced.
  - The customers can be viewed in great detail(i.e. **all columns**) or in concise style(i.e. **some** important **columns**).

- Four different kinds of command buttons are available on this screen; their functions are as follows:
  - **Back to Form View** Key: This key will lead the user back to the ***Record customers' bad check*** panel.
  - **Search the Query** Key: This key must be pressed after the user chooses to view a customer group's information (the selected criteria is based on the frequency of bounced checks).
  - **Print the Report** Key: Using this key, the system will send the customized Crystal Report to the user who needs it.
  - **Finish** Key: This key will terminate the program flow.

Access Any Screen(Command Key: Finish) →Ending Splash Screen



Note

- This is the very last splash screen of the DMDS software.

CHAPTER TEN

APPLICATION LIMITATIONS AND FUTURE DEVELOPMENT

APPLICATION LIMITATIONS

Even the best program has its limits. In fact, the larger and more complex a program is, the more restrictions it is likely to have. In this section, the Decision Making Database System(DMDS) limitations and performance will be addressed.

Since Visual Basic and Access are the underlying language and supporting database, respectively, for the DMDS, they inherit all the restraints of the Jet database engine which they both share with. The following tables are the four aspects (i.e., Database, Table, Form and Report, and Query) of the application's limitations, [9,18] which future developers should greatly concern themselves with:

Table 10.1 Database specifications

Attribute	Maximum
Database(.mdb) file size	1 gigabyte. Because database can include linked tables in other files, its total size is limited only by available storage capacity.
Number of objects in a database	32,768
Number of characters in an object name	64
Number of characters in a password	14
Number of characters in a user name or group name	20
Number of concurrent users	255

Table 10.2 Table specifications

Attribute	Maximum
Number of characters in a table name	64
Number of characters in a field name	64
Number of fields in a table	255
Table size	1 gigabyte
Number of characters in a text field	255
Number of characters in a Memo field	65,535
Size of an OLE object field	1 gigabyte
Number of indexes in a table	32
Number of fields in an index	10
Number of characters in a validation message	255
Number of characters in a table or field description	255
Number of characters in a validation rule	2,048
Number of characters in a record(excluding Memo and OLE object fields)	2,000
Number of characters in a field property setting	255

Table 10.3 Form and report specifications

Attribute	Maximum
Number of characters in a label	2,048
Number of characters in a text box	65,535
Form or report width	22 in. (55.87cm)
Section height	22 in. (55.87cm)
Height of all sections plus section headers(in Design view)	200 in. (508cm)
Number of levels of nested forms or reports	3
Number of fields or expressions one can sort or group on in a report	10
Number of headers and footers in a report	1 report header/footer; 1 page header/footer; 10 group headers/footers
Number of printed pages in a report	65,536

Table 10.4 Query specifications

Attribute	Maximum
Number of tables in a query	32
Number of fields in a recordset	255
Recordset size	1 gigabyte
Number of sorted fields in a query	10
Number of levels of nested queries	50
Number of characters in a cell in the query design grid	1,024
Number of characters for a parameter in a parameter query	255
Number of ANDs in a WHERE or HAVING clause	40
Number of characters in an SQL statement	approximately 64,000

In order to ensure that the DMDS use the Jet engine functions as intended, the software developers' or system engineers' must adjust the system configuration for the peak performance of the application. [19]Visual Basic, as an underlying application language, does provide a default setting for most common operations automatically. However, there are several approaches to tune these settings, such as PageTimeout, ReadAheadPages, and MaxBufferSize etc. So, determining the best setting for the DMDS can be difficult and time-consuming. In addition, an optimal setting for one operation may not be as good for others.

The intention of this report is not to enlist all the possible factors that affect the application performance, but to point out a guideline for the problems most users will likely encounter. As the application administrators attach the products and customer records to the system, more

and more memory is used in storing these records. Eventually, if the system runs out of space (about 2,000,000 records in a table) the system performance degradation (e.g. slow response time) will occur.

Generally, one should avoid such severe degradation by increasing RAM (Random Access Memory) on the computer system. The Decision Making Database System (DMDS) requires a minimum of 12 MB of RAM for Windows 95, but additional RAM improves performance. [19]The other tuning technique that might get better performance by specifying the available virtual memory is at least 25MB minus available space of RAM. For example, if one's computer system has 16 MB of RAM, then user should specify at least 9 MB of virtual memory. One should specify more if the DMDS is running a very large data set.

FUTURE DEVELOPMENT

The Decision Making Database System (DMDS), as it was presented in this project, is intended to show some of the capabilities of Visual Basic and Access for the data mart data structure. The DMDS introduced here is just one feasible starting point of the many implementations of data mart design. One may want to take the concepts presented in this project and extend them by:

- **Extending project scope to Internet possibility through ActiveX controls**

  [20]An ActiveX control is a stand-alone software component that does specific things in a standard way. The ActiveX controls have many newly developed features to make these controls much more usable with the Internet. Future development of this project can be greatly simplified by plugging one or more ActiveX controls into the DMDS application. This takes full advantage of existing component software functionality (the ActiveX has the largest number of components built in today's market), which is supplied by dozens of companies.

- **Creating Smarter Error-Handling Systems**

  Handling runtime errors is an essential part of any program. It is not enough for DMDS to just have several error-handling mechanisms that are scattered around. Creating smarter error-handling systems is knowing what to expect and how to handle the unexpected. More informative error messages and confirmation routines allowing users the chance to correct careless mistakes are all part of a smarter error-handling system.

- **Designing Multi-user Consideration**

  The challenges for the Multi-user application are database locking schemes and transaction management. It is necessary to fully utilize the Microsoft Jet database engine to provide three levels of locking scheme. This prevents two updates from occurring at the same time. [9] Another important feature that can add to the integrity of DMDS is using transactions operations to manage database update and delete maintenance routines[16].

- **Developing more solid system security and encryption work**

  One can develop routines that will, for example, record user activity, user login/logout history, and encrypt the users' passwords or other important information. These are just a few possibilities for the future edition of DMDS.

- **Preparing online detail help for the application**

  Including detail online help is a big plus to make an application have a finished appearance. Also, if such a resource is available, attaching online help will make the system easier to use and greatly enhance users' confidence and peace of mind.

CONCLUSIONS

In conclusion, the Decision Making Database System (DMDS) for Windows 95 was developed for the decision-makers' community. Object Linking and Embedding (OLE) technologies and Structure Query Language (SQL) queries were used for delivering fast, effective, and comprehensive business solutions. In this project, the DMDS was shaped up by adapting data mart Information Packaging Methodology's refinement technology. Such methodology begins with information package diagrams, evolves into star schemata, and then moves into the final physical implementation phase, using Visual Basic as coding language and Access as back-end database support.

Nevertheless, data mart implementations are still in a stage of immaturity. But since data mart carries the fullest potential to gain competitive advantages, it should be an important strategic tool in every corporation for the future perspective.

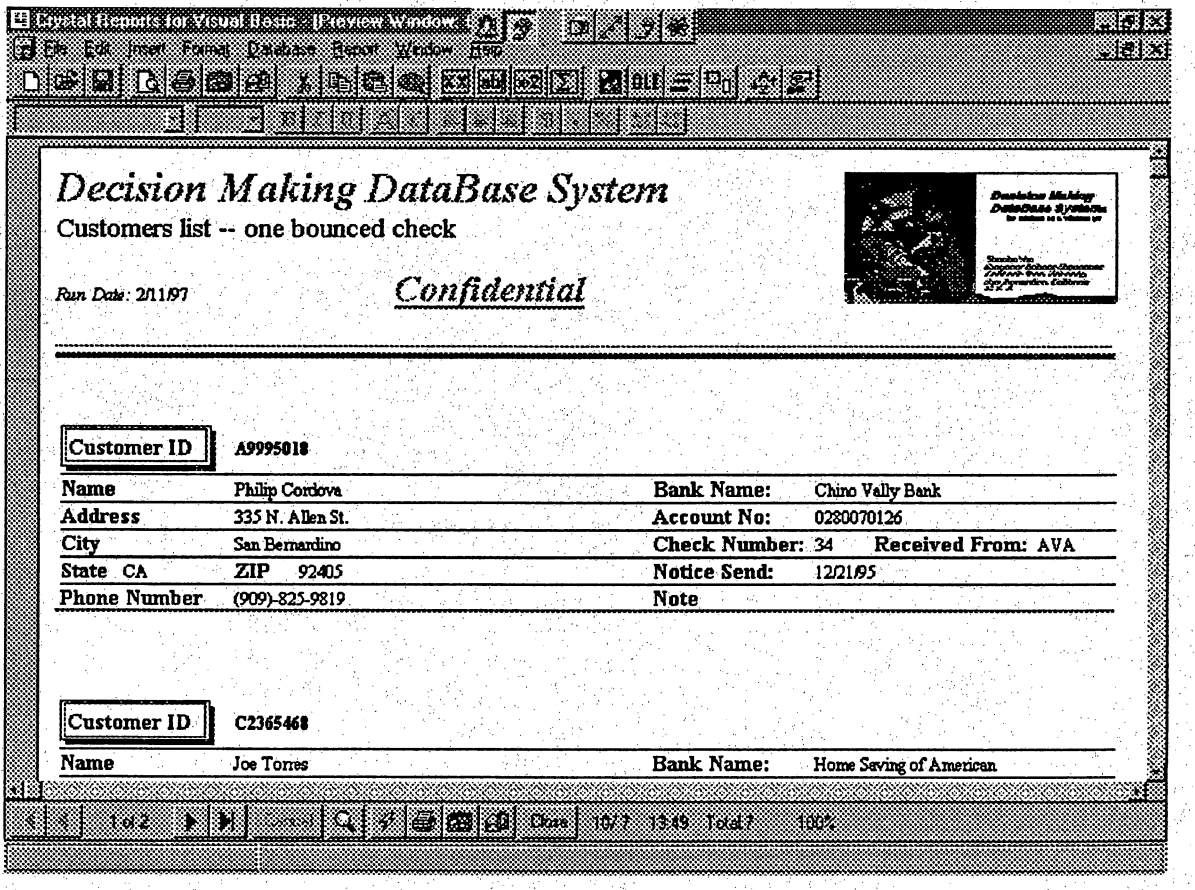
Moreover, my experiences in cultivating this project also offered me a chance to browse through other methodologies on delivering successful data marting strategies. Due to restricting on the length of the project, interested readers may refer to entries 21, 22, and 23 of the Bibliography for more information on how to apply

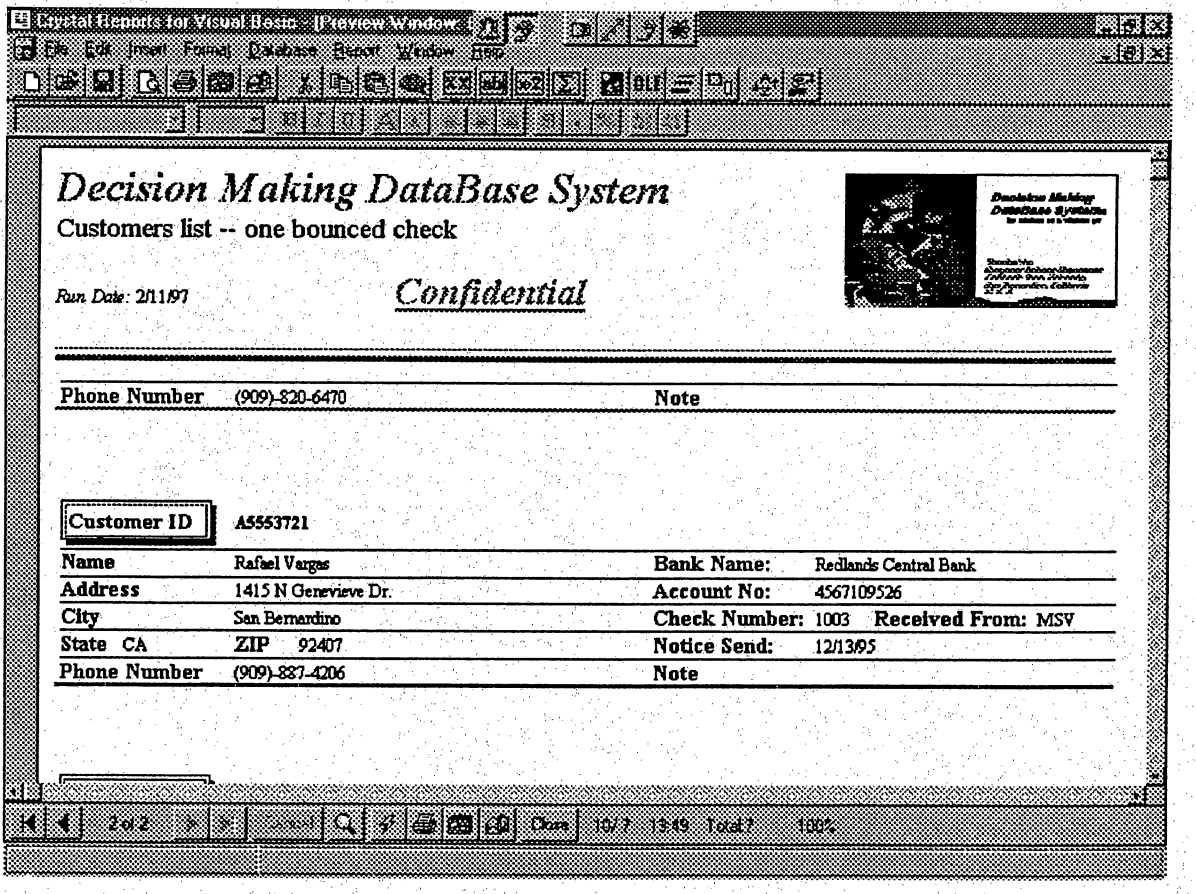
alternative methodologies to fine-tune the implementation techniques and move closer to the promising future of data mart.

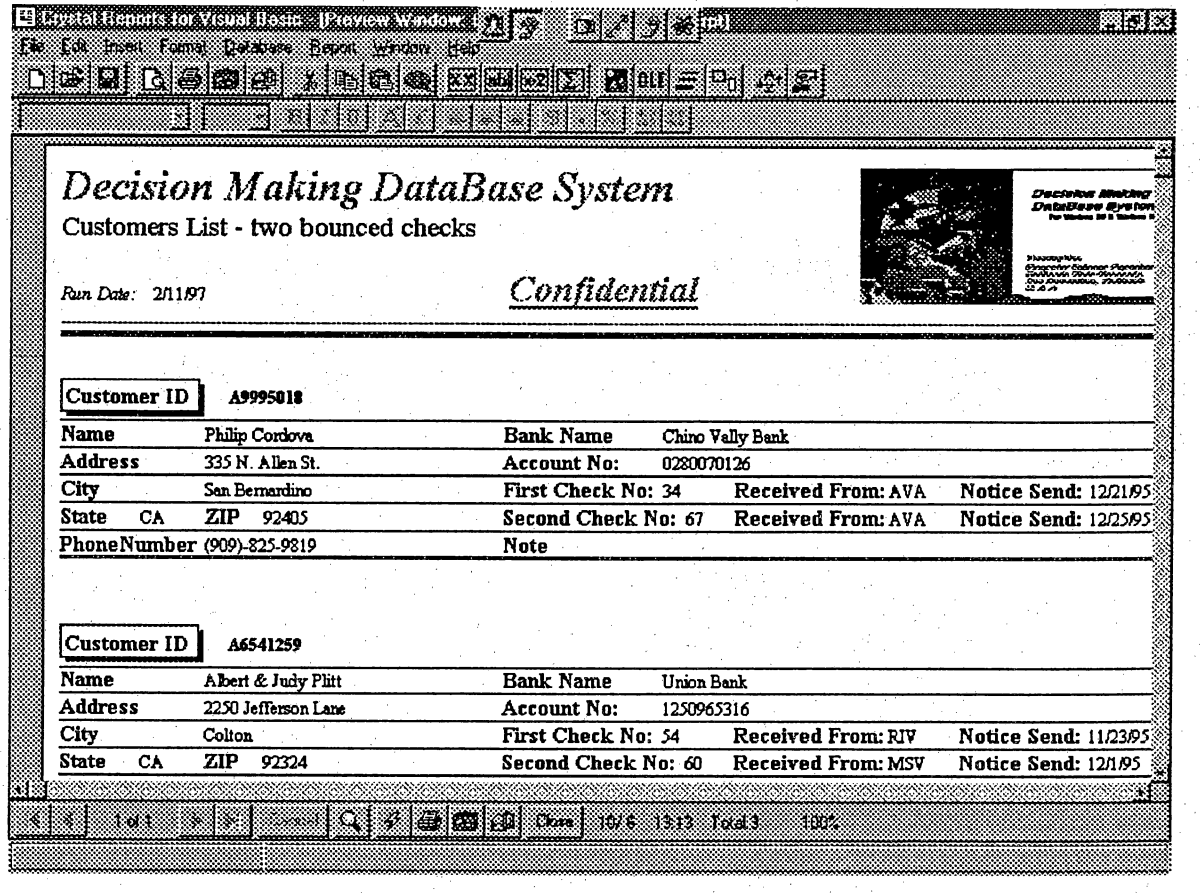
The source code of this project is available on the Internet with the following URLs:
http://web.csusb.edu/public/csci/swen, or
http://csnt.net.csusb.edu/projects/swen.

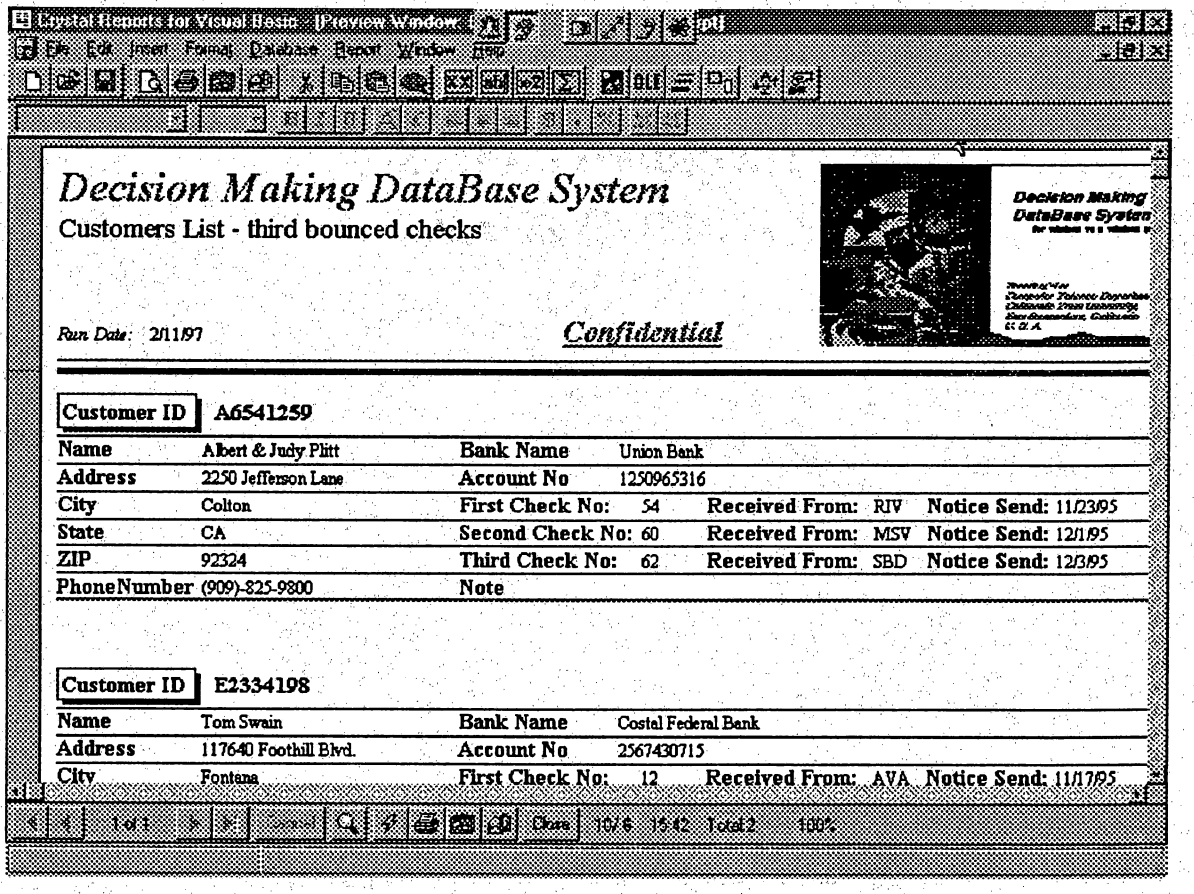
In addition, for the formalization of this project in the Unified Modeling Language (UML) Model interested individual, can greatly benefit from the work of Dr. Richard Botting which can be found at http://www.csci.csusb.edu/dick/samples/project.html. The information of Star schema recommended by Dr. Josephine Mendoza can be found at URL http://www.netmar.com/~nraden/str101.htm.

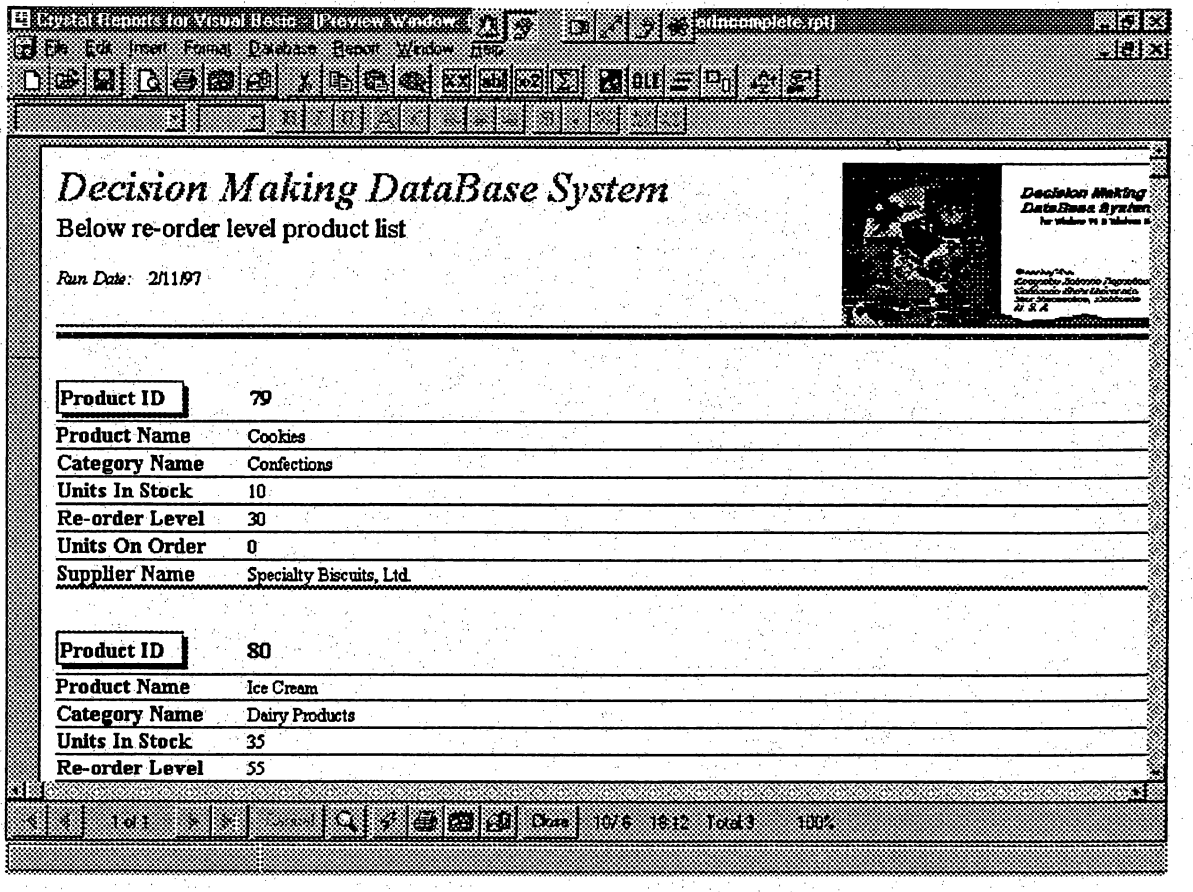
APPENDIX A:
REPORTS

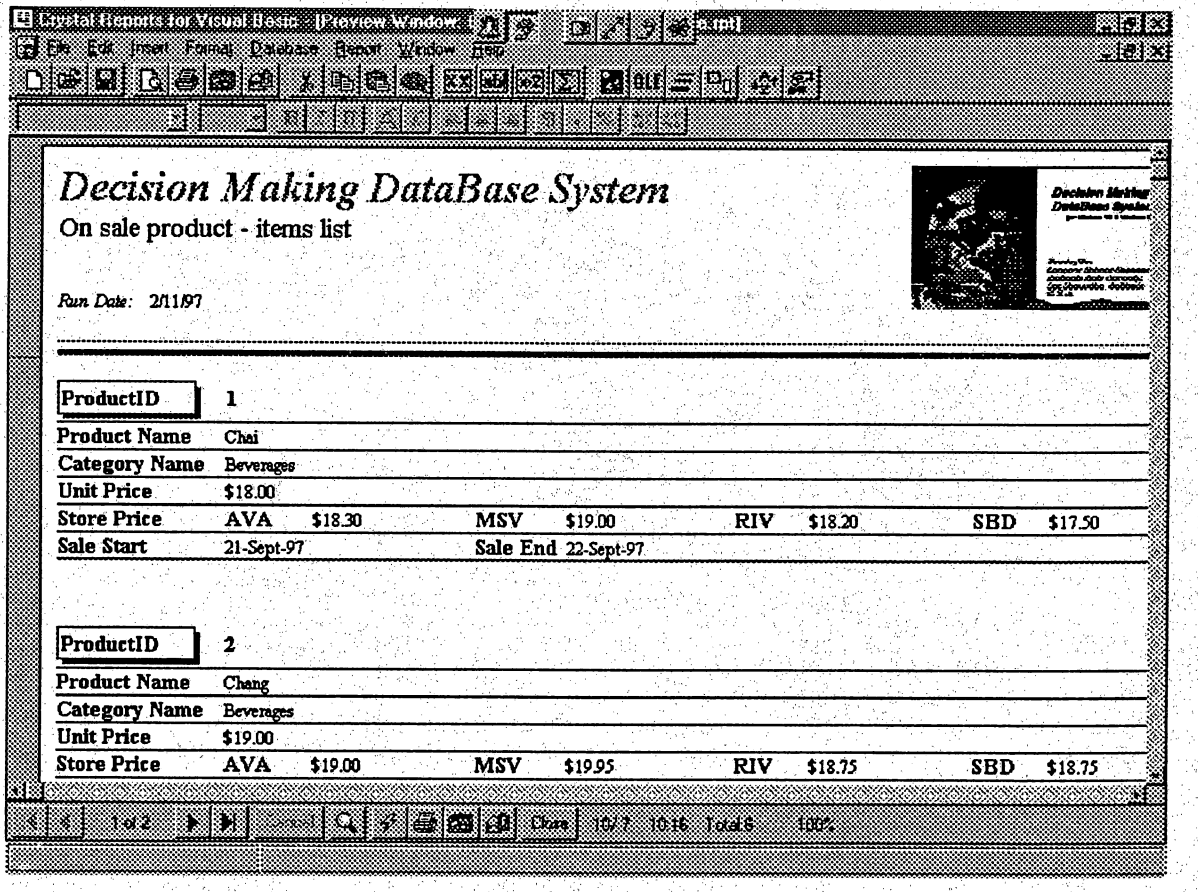


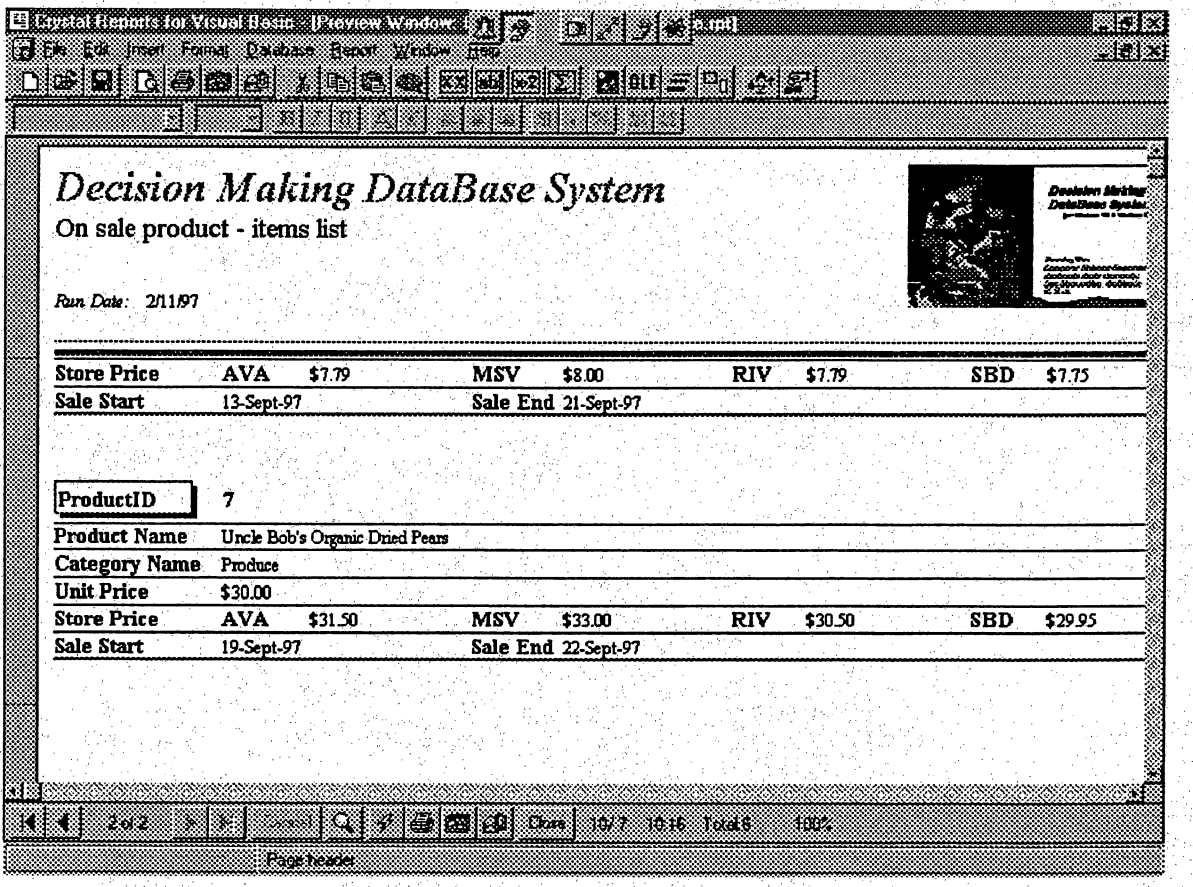

Crystal Reports for Visual Basic - [Preview Window 1]
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
Customers list -- one bounced check
Run Date: 2/11/97
Confidential
Phone Number (909)-820-6470 Note
Customer ID A5553721
Name Rafael Vargas Bank Name: Redlands Central Bank
Address 1415 N Genevieve Dr. Account No: 4567109526
City San Bernardino Check Number: 1003 Received From: MSV
State CA ZIP 92407 Notice Send: 12/13/95
Phone Number (909)-887-4206 Note
2 of 2
Done

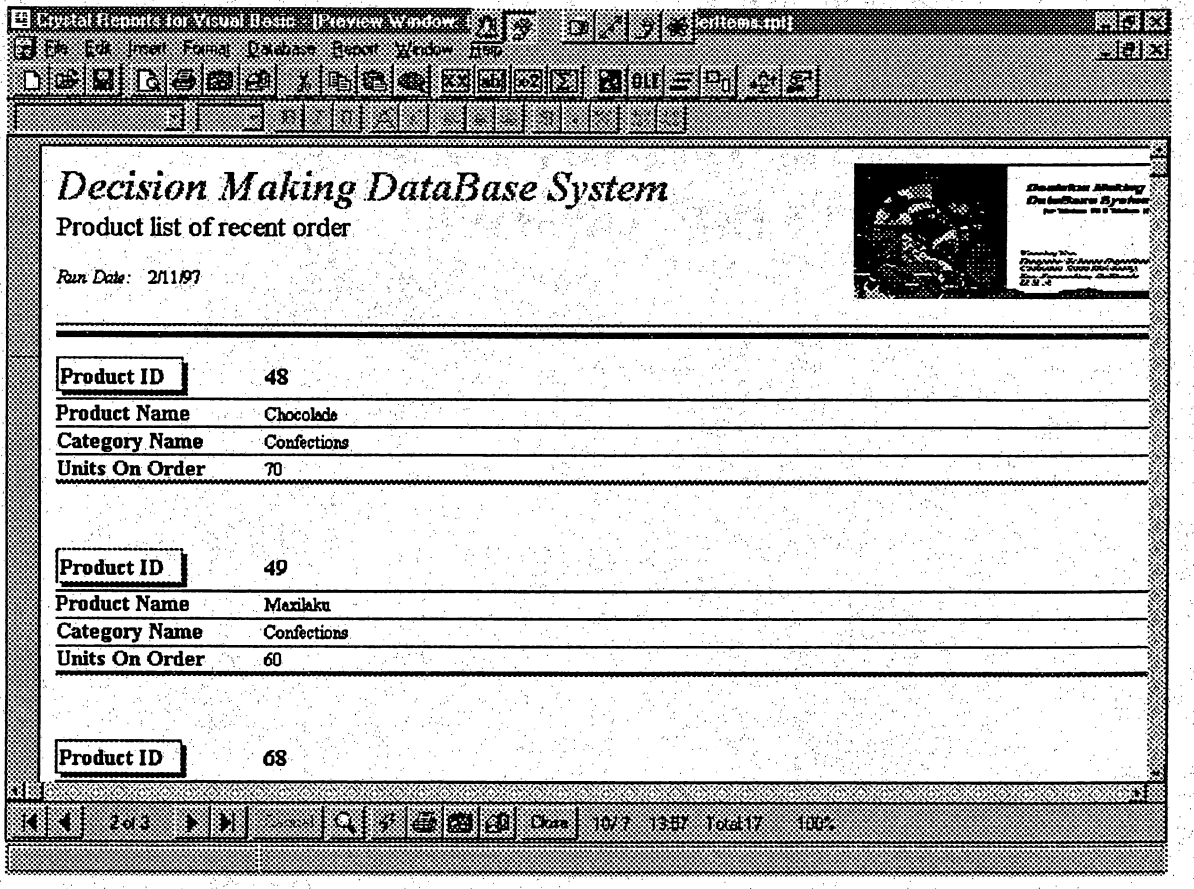

Crystal Reports for Visual Basic - [Preview Window]
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
Customers List - two bounced checks
Run Date: 2/11/97
Confidential
Customer ID A9995018
Name Philip Cordova
Address 335 N. Allen St.
City San Bernardino
State CA ZIP 92405
PhoneNumber (909)-825-9819
Bank Name Chino Valley Bank
Account No: 0280070126
First Check No: 34 Received From: AVA Notice Send: 12/21/95
Second Check No: 67 Received From: AVA Notice Send: 12/25/95
Note
Customer ID A6541259
Name Albert & Judy Plitt
Address 2250 Jefferson Lane
City Colton
State CA ZIP 92324
Bank Name Union Bank
Account No: 1250965316
First Check No: 54 Received From: RIV Notice Send: 11/23/95
Second Check No: 60 Received From: MSV Notice Send: 12/1/95
1 of 1
Close
Total 3
100%

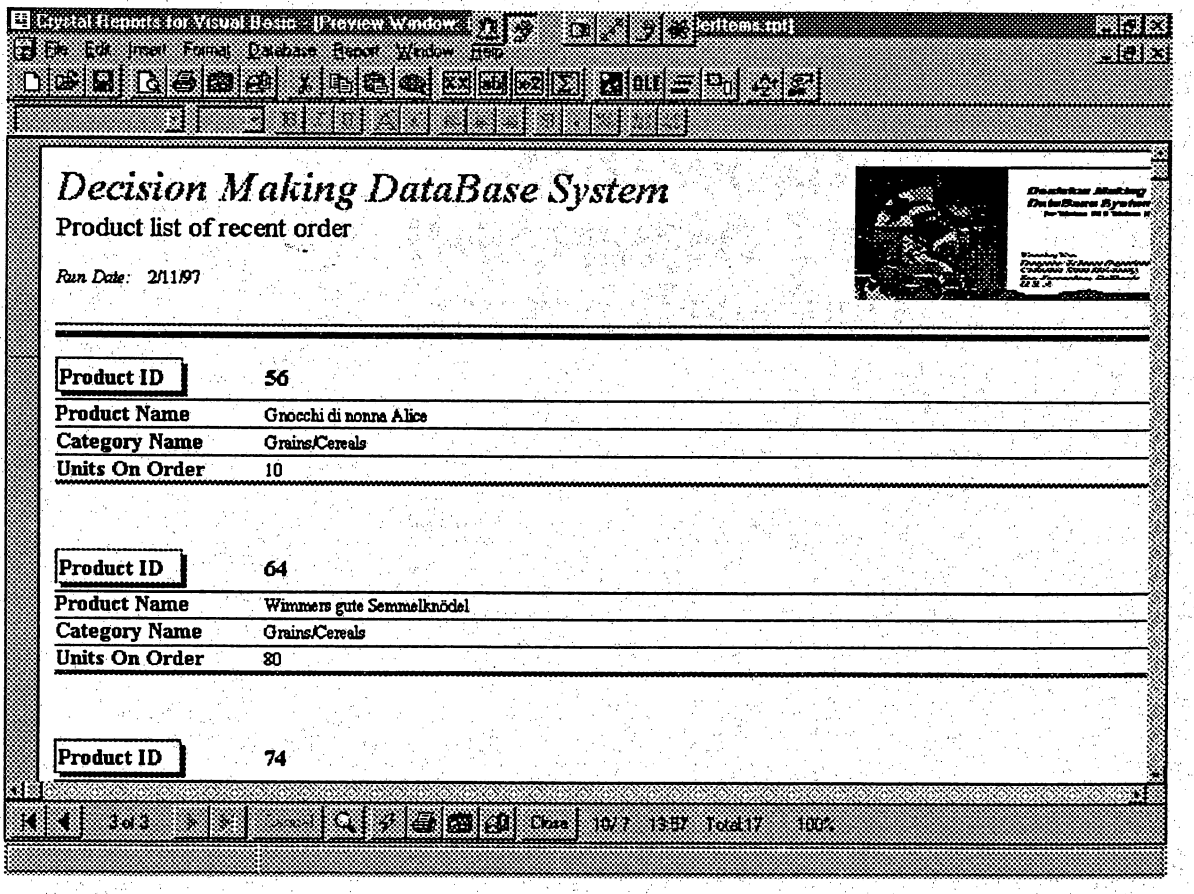

Crystal Reports for Visual Basic - [Preview Window 1]
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
Customers List - third bounced checks
Run Date: 2/11/97
Confidential
Decision Making DataBase System
Customer ID A6541259
Name Albert & Judy Plitt
Address 2250 Jefferson Lane
City Colton
State CA
ZIP 92324
PhoneNumber (909)-825-9800
Bank Name Union Bank
Account No 1250965316
First Check No: 54 Received From: RIV Notice Send: 11/23/95
Second Check No: 60 Received From: MSV Notice Send: 12/1/95
Third Check No: 62 Received From: SBD Notice Send: 12/3/95
Note
Customer ID E2334198
Name Tom Swain
Address 117640 Foothill Blvd.
City Fontana
Bank Name Costal Federal Bank
Account No 2567430715
First Check No: 12 Received From: AVA Notice Send: 11/17/95
1 of 1
Done
Total 2
100%


Crystal Reports for Visual Basic - [Preview Window1]
allincomplete.rpt
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
Below re-order level product list
Run Date: 2/11/97
Decision Making DataBase System
Product ID 79
Product Name Cookies
Category Name Confections
Units In Stock 10
Re-order Level 30
Units On Order 0
Supplier Name Specialty Biscuits, Ltd.
Product ID 80
Product Name Ice Cream
Category Name Dairy Products
Units In Stock 35
Re-order Level 55
1 of 1
Done
10/6 16:12 Total 3 100%


Crystal Reports for Visual Basic - [Preview Window 1]
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
On sale product - items list
Run Date: 2/11/97
ProductID 1
Product Name Chai
Category Name Beverages
Unit Price $18.00
Store Price AVA $18.30 MSV $19.00 RIV $18.20 SBD $17.50
Sale Start 21-Sept-97 Sale End 22-Sept-97
ProductID 2
Product Name Chang
Category Name Beverages
Unit Price $19.00
Store Price AVA $19.00 MSV $19.95 RIV $18.75 SBD $18.75
1 of 2


Crystal Reports for Visual Basic - [Preview Window 1]
File Edit Insert Format Database Report Window Help
Decision Making DataBase System
On sale product - items list
Run Date: 2/11/97
Store Price AVA $7.79 MSV $8.00 RIV $7.79 SBD $7.75
Sale Start 13-Sept-97 Sale End 21-Sept-97
ProductID 7
Product Name Uncle Bob's Organic Dried Pears
Category Name Produce
Unit Price $30.00
Store Price AVA $31.50 MSV $33.00 RIV $30.50 SBD $29.95
Sale Start 19-Sept-97 Sale End 22-Sept-97
2 of 2
Page header


Crystal Reports for Visual Basic - [Preview Window]
File Edit Insert Format Database Report Window Help
orders.mdb
Decision Making DataBase System
Product list of recent order
Run Date: 2/11/97
Product ID 48
Product Name Chocolade
Category Name Confections
Units On Order 70
Product ID 49
Product Name Maxilaku
Category Name Confections
Units On Order 60
Product ID 68
2 of 3
Done
10.7 13.97 Total 17 100%


Crystal Reports for Visual Basic - [Preview Window 1]
File Edit Insert Format Database Report Window Help
lerItems.rpt]
Decision Making DataBase System
Product list of recent order
Run Date: 2/11/97
Product ID 56
Product Name Gnocchi di nonna Alice
Category Name Grains/Cereals
Units On Order 10
Product ID 64
Product Name Wimmers gute Semmelknödel
Category Name Grains/Cereals
Units On Order 80
Product ID 74
3 of 3
Done
10/7 13:07 Total:17 100%

Appendix B:
Microsoft Jet Database Engine

As the previous chapters indicated, the primary database engine support of the Decision Making Database System (DMDS) is called the Jet Database Engine. The Jet in *Jet Database Engine* stands for Joint Engine Technology.

The Jet Database Engine is a set of components, generally referred to as DLLs (Dynamic Link Libraries). Such DLLs are designed to provide specific translation routines between requests and the target database in order to perform various functions. These specific translation routines are available for the native database - Microsoft Access - and also for other non-native ISAM (Indexed Sequential Access Method) databases in several popular formats, including Btrieve, dBase, FoxPro, Paradox, Excel and Lotus 1-2-3.

The Jet Database Engine is not made of a single program entity; it consists of three main components: the Jet Query Engine, the Internal ISAM and the Replication Engine. The task for the Jet Query Engine is to provide the translation of database SQL (Structured Query Language) statements and security services. The Internal ISAM handles for providing storing and retrieving access to external (non-native) data formats. The Replication Engine is responsible for creating

and periodically synchronizing the exact duplicates of a database at multiple system locations.

The detailed inner structures and associate functionalities of the Microsoft Jet Database Engine go beyond the scope of this project. If interested individuals want to learn more about the heart of this powerful Visual Basic database system engine, one can refer to entries 9, 14, 24, and 25 of the Bibliography or look through various online resources of the Microsoft corporation.

BIBLIOGRAPHY

[1] Eckerson, Wayne W., 'The Data mart option' *Open Information Systems,* vol. 11, no.7, July 1996, pp.3-16.

[2] Goldberg, P., 'Guidelines for defining requirements for decision support systems' *Data Resource Management Journal,* spring 1991.

[3] Comaford, Christine, 'Attention, data mart shoppers!' *PC week,* vol. 13, no. 8, September. 1996, p.61.

[4] Demarest, Marc, 'Building the data mart' *DBMS*, vol. 7, no. 8, July 1994, pp.44-51.

[5] Laplante, Alice, 'Big things come in smaller packages' *ComputerWorld*, vol. 30, no. 26, June 24, 1996, pp.88-89.

[6] Gurton, Annie, 'Same song, different tune', *Computer Weekly*, July 1996, p.40.

[7] Inmon, W. H., 'Data structures in the information warehouse' *Enterprise Systems Journal*, January 1992.

[8] Inmon, W. H., *Building the Operational Data Store*, John Wiley & Sons, Inc., 1996.

[9] Haught, Dan and Ferguson, Jim, *Microsoft Jet Database Engine Programmer's Guide*, Microsoft Corp.,1997.

[10] *Microsoft Visual Basic Programmer's Guide*, version 4.0, Microsoft Corp., 1995.

[11] Hammergren, Tom, *Data Warehousing Building the Corporate Knowledgebase*, International Thomson Computer Press, 1996.

[12] Marshall, Morwenna, 'Data mart migration made simple' *Database Programming & Design*, vol.9, no.8, August 1996, p. 68.

[13] Bruce, Walter R., Madoni, Dan, and Wolf, Rich, *The visual guide to Microsoft Access for Widows 95: the pictorial companion to Windows database management & programming*, Ventana Communications Group, Inc., 1995.

[14] Roof, Larry, *The Revolutionary Guide to Visual Basic 4 Professional*, Wrox Press, 1996.

[15] Heyman, Mark S., *Essential Visual Basic 4*, Sam Publishing, 1995.

[16] Jennings, Roger, *Database Developer's Guide with Visual Basic*, Sam Publishing, 1996.

[17] Pressman, Roger S., *Software Engineering: A Practitioner's Approach*, McGraw-Hill Book Company, 1982.

[18] Microsoft Access for Windows 95, version 7.0, Online manual.

[19] *Info: Perform.txt: Performance Tuning Tips for VB and Access*, SupportOnline, Microsoft Technical Support, Article ID: Q107751. September 29, 1997.

[20] Chappell, David *Understanding ActiveX and OLE*, Microsoft Press, 1996.

[21] Kimball, Ralph *The Data Warehouse Toolkit*, John Wiley & Sons, Inc., 1996.

[22] Gill, Harjinder S. and Rao, Prakash C. *The Official Client/Server Computing Guide to Data Warehousing*, Que Corporation, 1996.

[23] Inmon, W. H. *Building the Data Warehouse*, John Wiley & Sons, Inc., 1996.

[24] Roman, Steve *Access Database Design & Programming*, O'Reilly & Associates, Inc., 1997.

[25] *Microsoft Visual Basic Guide to Data Access Objects and Crystal Reports for Visual Basic*, Microsoft Corp., 1997.